Audio-fingerprints and associated indexing strategies for the purpose of large-scale audio-identification

Sebastien Fenet

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0051

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « SIGNAL et IMAGES »

présentée et soutenue publiquement par

Sébastien FENET

le 23 septembre 2013

Empreintes Audio et Stratégies d'Indexation Associées pour l'Identification Audio à Grande Echelle

Audio-Fingerprints and Associated Indexing Strategies for the Purpose of Large-Scale Audio-Identification

Directeur de thèse : **Gaël RICHARD**
Co-directeur de thèse : **Yves GRENIER**

Jury

M. Frédéric BIMBOT, Directeur de Recherche, IRISA, Rennes	Président
Mme Régine ANDRE-OBRECHT, Professeur, IRIT, Université Paul Sabatier, Toulouse	Rapporteur
M. Sylvain MARCHAND, Professeur, Université de Bretagne Occidentale, Brest	Rapporteur
M. Geoffroy PEETERS, Chargé de Recherche HDR, IRCAM, Paris	Examinateur
M. Avery WANG, Founder et Chief Scientist, Shazam Entertainment, Palo Alto USA	Examinateur

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Audio-Fingerprints and Associated Indexing Strategies for the Purpose of Large-Scale Audio-Identification

Sébastien Fenet

Abstract

Throughout this work, we explore various audio-fingerprint models associated with indexed-based search strategies for the purpose of large scale audio-identification.

We start with the description of the audio-identification use-case, which consists of automatically retrieving the meta-data associated to an unknown sound. Audio-fingerprint algorithms meet this objective by extracting from the audio signals a characteristic fingerprint. By learning beforehand all the fingerprints of a set of references, the algorithm is then able to identify any signal that belongs to this set. This is done by extracting the fingerprint from the unknown signal and looking for the closest learnt fingerprint. The main stakes when designing an audio-fingerprint algorithm are the scalability and the robustness to distortion. The algorithm must indeed be able to manage a very large set of references (typical industrial databases include hundreds of thousands of music titles). Besides, the algorithm must be able to identify any signal that corresponds to a reference, even if it has undergone a series of distortions. As far as the distortions are concerned, we make a specific distinction between post-processing distortions, which are the ones that occur in the transmission channel of a given music recording (dynamic compression, equalisation, pitch-shifting, additional noise...) and the variations that occur when we study two different recordings of one same music title. In the first case, we talk about *exact matching* whereas we use the terminology *approximate matching* in the second case.

We subsequently propose an exhaustive study of the state of the art in audio-fingerprint. Amongst the methods from the state of the art, we propose a specific focus on the one proposed in [Wan03], commonly referred to as "Shazam's method". We give a detailed explanation of the method and finally demonstrate that it lacks robustness to the pitch-shifting distortion. Our third part is therefore dedicated to the proposition of improvements over our initial implementation of [Wan03]. These include the proposition of different signal models for the fingerprint and the adaptation of the method to an extended functional perimeter.

In the last part of the work, we focus on a quite different issue. We start from the observation that, to our knowledge, virtually no method from the state of the art in audio-fingerprinting deals with the approximate matching use-case. We thus propose two novel audio-fingerprint methods that meet the objectives of approximate matching. The first approach is based on the transcription of the signal in a chords sequence. The second one relies on an innovative representation of the signal. The latter is composed of compact states that include both rhythmic and harmonic information.

Contents

Part I

Extended Summary (in French)

Chapter 1

Qu'est-ce que l'Audio-Fingerprint

1.1 Principes

L'identification audio par le contenu consiste à retrouver des méta-données (telles que, pour une musique : le titre, l'artiste, l'album...) associées à un extrait audio. L'identification audio a attiré l'attention de la communauté scientifique durant la dernière décade du fait de son rôle clé dans un nombre important d'applications [CBG+02], les deux plus célèbres étant l'identification d'un extrait audio à travers le réseau de téléphone mobile et la surveillance automatique de contenu protégé.

L'identification audio par extraction d'empreinte (ou encore audio-fingerprint) consiste à extraire du signal audio une empreinte. Celle-ci permet d'identifier de manière univoque le signal qui peut alors être associé à ses méta-données. Plus précisément, le système dispose d'une base de données de références (en pratique, ce sont des fichiers audio). Les références correspondent à l'ensemble des signaux audio que le système sera en mesure d'identifier. Le principe de l'audio-fingerprint est de ramener la comparaison de signaux audio à celle de leurs empreintes. Ainsi, une base d'empreintes de références est créée à partir de la base de références audio lors de la phase d'apprentissage (voir figure 1.1). Lorsque le système identifie un extrait inconnu, il calcule son empreinte et cherche ensuite l'empreinte la plus proche dans la base d'empreintes (voir figure 1.2).

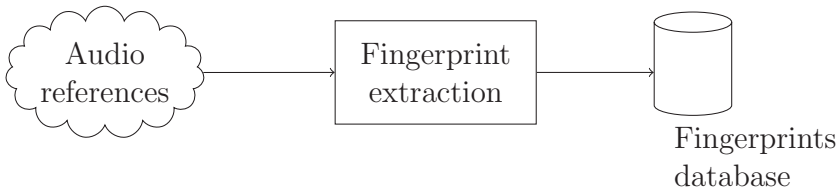


Figure 1.1: Apprentissage des empreintes de référence par l'algorithme d'audio-fingerprint.

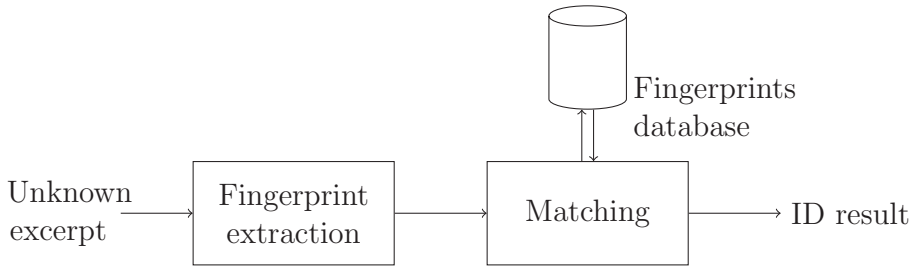


Figure 1.2: Identification d'un extrait inconnu par l'algorithme d'audio-fingerprint.

1.2 Enjeux

1.2.1 Robustesse aux Distorsions

L'empreinte doit idéalement permettre d'identifier chaque référence de manière univoque. Une difficulté supplémentaire, en identification audio, est que l'on s'emploie à mettre en œuvre des représentations qui sont invariantes par un certain nombre de distorsions du signal. On conçoit dès lors la difficulté du domaine qui consiste à trouver un compromis entre une empreinte suffisamment univoque pour permettre une identification correcte et une empreinte suffisamment "souple" pour être insensible aux distorsions.

La robustesse aux distorsions couvre en fait deux situations qui, bien qu'elles peuvent sembler parfois très proches pour l'auditeur humain, sont très dissimilaires pour une machine.

- Dans le premier cas, l'idée est d'identifier un extrait audio qui est une copie exacte d'une référence. Le signal à identifier a cependant pu subir un certain nombre de distorsions qui se produisent dans la chaîne de traitement. Nous parlons alors de distorsions en *post-traitement* ; les plus communément traitées dans les travaux en audio-fingerprint sont présentées dans le tableau 1.1. Ce premier cas d'usage est illustré en figure 1.3 et nous l'appelons *identification exacte*.

- Dans le deuxième cas, l'extrait audio n'est pas explicitement dans la base de références. Cependant, la base contient une référence qui est musicalement très proche du signal à

Figure 1.3: Illustration de l'*identification exacte*. L'algorithme doit associer deux signaux qui correspondent au même enregistrement avec différents post-traitements.

identifier. Typiquement, la base contient un enregistrement studio d'un titre donné et le système doit identifier une version *live* du même titre. Nous disons que les signaux sont *similaires* et nous parlons d'*identification approchée*, illustrée en figure 1.4.

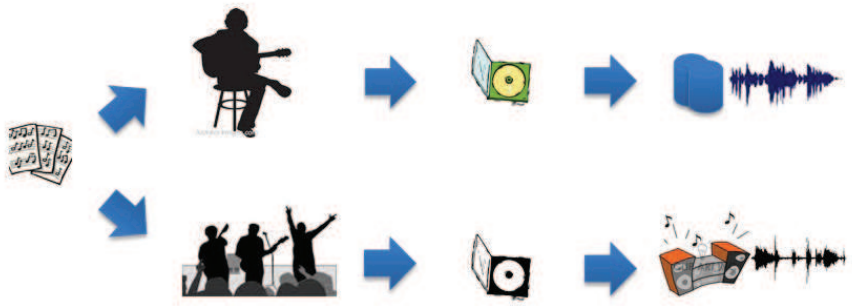


Figure 1.4: Illustration de l'*identification approchée*. L'algorithme doit associer deux signaux qui correspondent au même titre, exécuté dans des conditions différentes (arrangement différent, musiciens différents, conditions d'enregistrement différentes, ...).

1.2.2 Passage à l'échelle

L'autre enjeu de l'audio-fingerprint est le passage à l'échelle. Dans le contexte des applications industrielles, les bases de références contiennent typiquement des centaines de milliers de titres, voire des millions. On peut notamment citer les bases utilisées par les sociétés de surveillance de copyright ou encore les initiatives open-source (Music Brainz, Echo Nest) dont les catalogues atteignent dorénavant le million de titres. La capacité d'un système à exécuter une recherche efficace est donc au cœur du problème. Afin d'atteindre de telles possibilités de passage à l'échelle, de nombreux travaux du domaine proposent une empreinte

Distorsion	Formule
Troncature	$\tilde{S} : \begin{array}{rcl} [0;M] \times [0;f_s/2] & \longrightarrow & \mathbb{C} \\ (t,f) & \longmapsto & S(t+t_0,f) \end{array}$
Egalisation	$\tilde{S}(t,f) = S(t,f).h(f)$ with $h : [0;f_s/2] \longrightarrow [0;1]$
Compression dynamique	$\tilde{S}(t,f) = S(t,f).g(t)$ with $g : [0;L] \longrightarrow [0;1]$
Glissement fréquentiel	$\tilde{S}(t,f) = S(t,Kf)$
Etirement temporel	$\tilde{S}(t,f) = S(K't,f)$
Ajout de bruit	$\tilde{S}(t,f) = S(t,f) + n(t,f)$

Table 1.1: Distorsions en post-processing traitées dans les cas d'utilisation traditionnels d'audio-fingerprint.

qui s'intègre dans un mécanisme d'indexation. L'indexation est en effet un outil classique dans le domaine des recherches à très grande échelle. Nous le présentons en détail dans la section suivante.

1.3 Indexation

L'indexation consiste à lister des caractéristiques d'objets d'une collection donnée. Chaque caractéristique référencée est associée à une liste contenant tous les objets qui possèdent ladite caractéristique. Dans la suite, nous appelons les caractéristiques listées les *clés* (ou *clés d'indexation*) et, pour indiquer l'association entre la caractéristique et les objets la possédant, nous disons que *la clé pointe vers les objets*.

Pour bien comprendre la puissance de recherche que procurent les techniques d'indexation, il est assez parlant de se référer au cas de la recherche dans un livre. Lorsque l'on cherche une page contenant un mot précis au sein d'un ouvrage, l'approche en "force brute" consiste à parcourir tout le livre jusqu'à trouver la page recherchée. En revanche, si un index est proposé à la fin du livre, il suffit de rechercher le mot au sein de l'index. Le lecteur dispose alors instantanément des numéros de pages contenant le terme recherché. Il suffira alors d'étudier ces pages *candidates* pour retrouver la page d'intérêt.

Nous pouvons noter que pour une même collection d'objets, il y a plusieurs façons de construire un index. En fonction de la stratégie retenue, l'efficacité de l'index peut radicalement changer. Par exemple, la table des matières d'un livre est également une forme d'index. Cet index, cependant, a une efficacité plus réduite puisqu'il ne référence que les termes utilisés dans les titres de chapitre ou de paragraphe du livre.

Les avantages d'un index ayant été exposés, nous comprenons l'intérêt de ces stratégies

pour le problème de l'audio-fingerprint. La difficulté, cependant, est que les index sont des processus qui sont exacts par nature. Imaginons que le lecteur cherche dans l'index le mot 'Barkov' au lieu du mot 'Markov', il semble clair que sa recherche n'aboutira pas. Ce problème doit être traité précautionneusement, particulièrement dans le cadre du travail avec des caractéristiques audio qui sont relativement versatiles. A travers l'exploration des différents modèles d'empreintes de notre travail, la question centrale sera donc constamment : *que pouvons-nous définir comme clé d'indexation dans notre modèle ?* Les questions périphériques qui vont avec sont : comment construisons-nous les requêtes à l'index ? comment traitons-nous les réponses de l'index ?

Chapter 2

Cas d'Usage et Protocole d'Evaluation

2.1 Cas d'Usage

Les spécifications du cas d'usage ont été faites dans le cadre du projet Quaero [1] et sont basées sur la "surveillance de diffusions multimédia". Le principe est la mise en place d'un système qui surveille de manière continue des flux multimédia (TV, radio, web...). Le système a appris au préalable une base de sons de référence (typiquement des musiques copyrightées). La tâche du système est alors de signaler toute diffusion d'un son de la base au sein du flux. Notons que, tel qu'indiqué en section 1.2.1, le son a pu subir un certain nombre de distorsions. Par ailleurs, il est intéressant de mentionner que le flux contient des sons de la base mais également d'autres portions de flux qui ne sont pas référencées.

Si nous désignons par $m_1, m_2, ..., m_N$ les références, par $\tilde{m}_1, \tilde{m}_2, ..., \tilde{m}_N$ leurs versions diffusées (et distordues) et par n le reste du flux, la tâche peut être illustrée comme en figure 2.1.

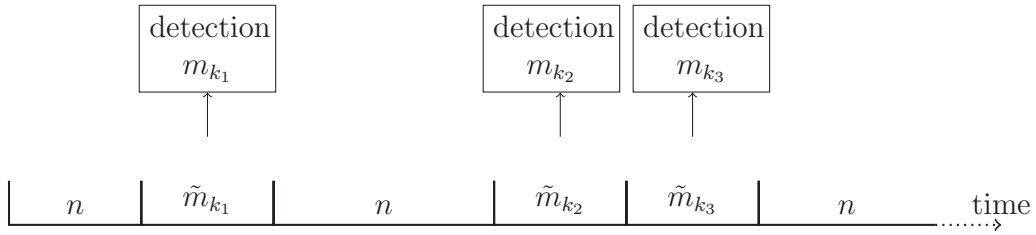


Figure 2.1: Scénario de surveillance de flux. Le système doit détecter dans un flux multimédia toutes les diffusions de sons appartenant à une base de référence.

[1] http://quaero.org

2.2 Protocole d'Evaluation

Avant chaque évaluation, la base du système est populée avec N références qui proviennent d'une base de données industrielle actuellement utilisée en production. Lesdites références sont des extraits de 60s de titres musicaux, chacune d'entre elles possédant un identifiant unique m_i.

Le corpus à analyser correspond à des journées de radio françaises. Afin de proposer un corpus avec un ensemble de distorsions le plus représentatif possible, il est bon de diversifier les sources (radios jeunes, classiques, rock, ...).

L'algorithme doit alors analyser les flux radio et signale toute détection d'item appartenant à la base de références. Pour chaque son de référence détecté, l'algorithme évalué génère une sortie qui spécifie l'identifiant de la référence détectée ainsi que la date de détection dans le flux.

Par ailleurs, Quaero nous fournit la vérité terrain, générée "à la main" par des annotateurs professionnels. Ces fichiers contiennent, pour chaque diffusion d'un titre musical, une annotation qui stipule l'identifiant de la référence ainsi que la date de début et la date de fin.

Nous proposons dans ce travail un score à deux composantes. La première composante est le ratio de détections correctes. Les titres à détecter sont ceux listés dans la vérité terrain. Chaque diffusion d'un titre de référence est annotée avec un identifiant m_{gt}, une date de début d_{gt}^{start} et une date de fin d_{gt}^{end}.

Definition 1 *Un titre diffusé est correctement détecté si : il existe au moins une sortie de l'algorithme dont l'identifiant m_{out} et la date de détection d_{out} sont tels que :*

$$\begin{cases} m_{gt} = m_{out} \\ d_{gt}^{start} \leq d_{out} \leq d_{gt}^{end} \end{cases} \tag{2.1}$$

Une bonne pratique consiste à écrire le ratio de détections correctes sous sa forme fractionnaire :

$$\frac{\text{Nombre de titres correctement détectés}}{\text{Nombre de titres à détecter}}$$

De cette façon, le lecteur a instantanément une idée de la taille du corpus.

La seconde composante est le nombre de fausses alarmes. Chaque sortie de l'algorithme est définie par un identifiant m_{out} et une date de détection d_{out}.

Definition 2 *Une détection est un vrai positif si : il existe une annotation dans la vérité terrain dont les identifiant m_{gt}, date de début d_{gt}^{start} et date de fin d_{gt}^{end} sont tels que :*

$$\begin{cases} m_{gt} = m_{out} \\ d_{gt}^{start} \leq d_{out} \leq d_{gt}^{end} \end{cases} \tag{2.2}$$

Toutes les sorties qui ne sont pas des *vrais positifs* sont des *fausses alarmes*. La seconde composante du score est simplement le nombre absolu de fausses alarmes.

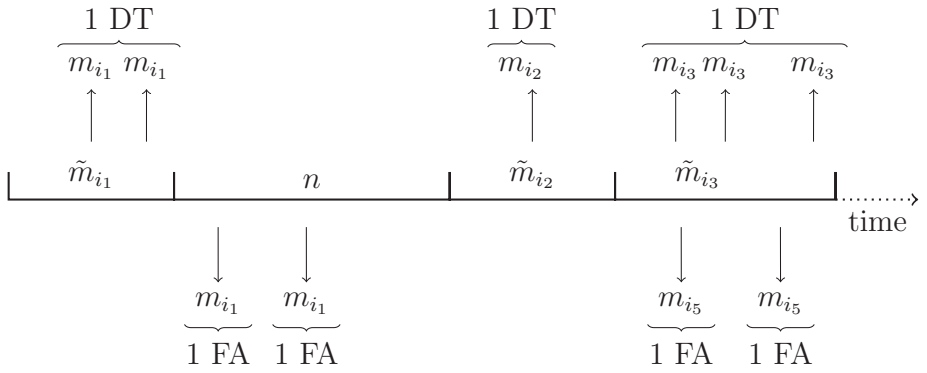


Figure 2.2: Illustration de la procédure de notation. Si une sortie (représentée par une flèche) contient l'identifiant m_i et a une date de détection qui est comprise entre la date de début et la date de fin annotées d'une occurrence du titre m_i, la diffusion est comptée comme un *titre détecté* (DT). Plusieurs détections de la même occurrence d'un titre sont comptées une seule fois (comme indiqué par les accolades horizontales). A l'inverse, si l'algorithme détecte une référence durant une plage vide (représentée par n) ou pendant une plage qui contient un autre titre, nous comptons une fausse alarme (FA). Il n'y a pas d'intégration sur les fausses alarmes : chaque fausse alarme est additionnée.

Chapter 3

Focus sur la Méthode Shazam

3.1 Introduction

Une des plus célèbres applications de l'audio-fingerprint a été développée par l'entreprise Shazam. Le travail de Shazam a été cité dans de nombreuses publications du domaine. Plus précisément, les auteurs citent en général l'article ISMIR de 2003 publié par Wang [Wan03]. La méthode qui y est décrite est souvent désignée par "méthode de Wang" ou "méthode de Shazam". Cet article constitue la seule publication de conférence au sujet de leur travail en audio-fingerprint. Il faut cependant noter qu'en dehors de cet article, Shazam a déposé plusieurs brevets [LcWC09, LcWSI11]. De plus, s'agissant d'une entreprise, il y a potentiellement toujours des différences entre les programmes réellement exécutés sur leurs machines et ce qu'ils laissent éditer de manière publique. A l'instar de la majorité des auteurs, nous nous concentrons sur le travail présenté dans l'article [Wan03]. Dans la suite, nous y faisons référence en tant que la méthode $Shazam^{03}$.

3.2 Principes

3.2.1 Calcul de l'Empreinte

La méthode $Shazam^{03}$ propose, pour le calcul de l'empreinte, de partir du spectrogramme du signal. L'idée est ensuite de binariser ce spectrogramme en mettant au niveau '1' les points d'énergie localement maximale et au niveau '0' les autres points. Une des exigences de la méthode est par ailleurs d'assurer une répartition homogène, en temps et en fréquence, des points à '1'.

Dans le but d'obtenir cette représentation binaire homogène, nous proposons une stratégie originale, simple et efficace qui consiste à recouvrir le spectrogramme d'un pavage régulier.

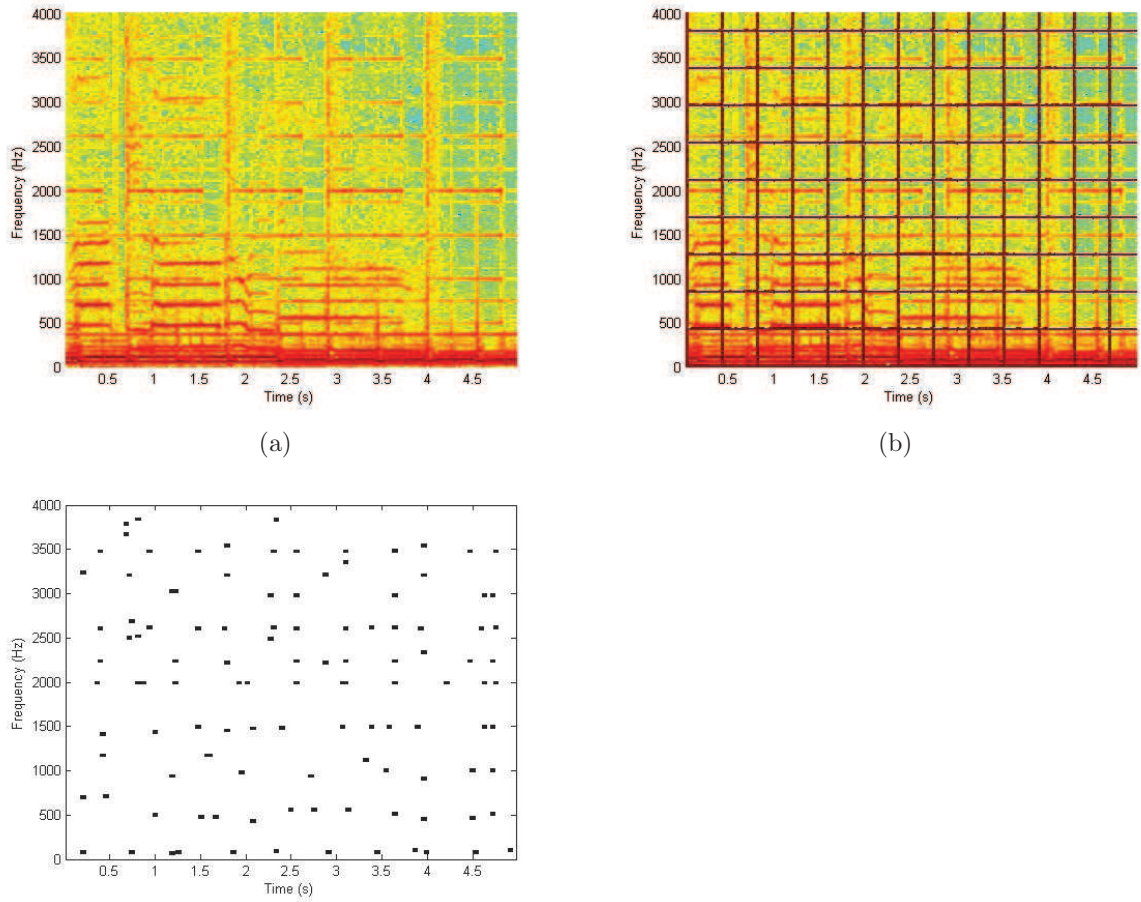


Figure 3.1: Calcul du spectrogramme binaire (c) proposé dans *Shazam*[03] grâce à un pavage (b) du spectrogramme original (a). Dans chaque tuile du pavage, le point d'énergie maximale est mis à '1' et les autres à '0'.

Dans chaque tuile du pavage, le point d'énergie maximale est mis à '1' et les autres à '0' (voir figure 3.1).

3.2.2 Stratégie de Recherche

Lorsque le système doit analyser un extrait inconnu, le système calcule son empreinte. La stratégie d'identification consiste alors à chercher, parmi les empreintes de référence, celle qui a le plus de points à '1' en commun avec l'empreinte inconnue. Afin de pouvoir parcourir rapidement de larges bases, Wang suggère la mise en place d'une stratégie d'indexation. Les clés d'indexation proposées par Wang correspondent à des paires de points à '1'.

Plus précisément, lors de la phase d'apprentissage, toutes les paires de points à '1' de chaque référence sont indexées. Lorsqu'une paire est stockée comme clé dans l'index, elle est

associée à l'identifiant de la référence la contenant, ainsi qu'à la date d'occurrence de cette paire dans la référence.

Lors de la phase d'identification, les paires de l'extrait inconnu sont calculées. Puis, chaque paire est utilisée comme requête à l'index. Celui-ci renvoie, à chaque fois, la liste des références contenant la paire et les dates d'occurrence. En étudiant la corrélation temporelle entre les occurrences des clés dans l'extrait inconnu et leurs occurrences dans les références renvoyées par l'index, il est facilement possible de trouver la référence ayant le plus de paires en commun avec l'extrait inconnu, à décalage temporel constant. Cette référence constituera le meilleur candidat à l'identification.

Notons que ce procédé requiert une stratégie d'encodage des clés (le système doit pouvoir représenter chaque paire de points). L'encodage proposé par Wang pour une paire de points de coordonnées (t_1, f_1) et (t_2, f_2) est le vecteur : $[f_1, f_2, t_2 - t_1]$.

3.2.3 Seuil de Détection

La stratégie décrite précédemment renvoie un meilleur candidat pour chaque requête. En revanche, il n'est pas, à ce stade, possible de savoir s'il s'agit d'une réelle identification ou d'une fausse alarme. Nos expériences ont montré que la mise en place d'un simple seuil sur le nombre de clés en commun entre le meilleur candidat et l'extrait inconnu ne donnait pas de bons résultats en termes de performance de détection.

Nous avons donc proposé un mécanisme de détection, basé sur la fusion de décisions locales, permettant de prendre une décision fiable quant à la correspondance entre le meilleur candidat et la requête. Globalement, la stratégie consiste à diviser la requête en plusieurs sous-requêtes et à identifier chacune de ces sous-requêtes. Si les meilleurs candidats obtenus pour les différentes sous-requêtes se correspondent en majorité (même référence avec un décalage temporel cohérent), nous considérons qu'il y a identification. Dans le cas contraire, nous considérons que l'extrait à identifier n'est en fait pas dans la base.

3.3 Expériences et Résultats

Rappelons que le protocole d'évaluation consiste à analyser des flux radiophoniques provenant de stations de radio réelles. L'algorithme doit détecter les diffusions des titres musicaux qui sont dans la base de références. Dans cette expérience, la base est composée de 7300 extraits de 60s de titres musicaux et le flux analysé correspond à 7 jours de la radio française 'RTL'. Comme les références de la base ne proviennent du flux radiophonique analysé, elles n'ont pas subi les mêmes post-traitements.

$Quaero_{detec}$	Detected titles / Total	False Alarms
$Shazam^{03}$ [Wan03]	381 / 459 (=83.0%)	0

Table 3.1: Résultats de notre implémentation de $Shazam^{03}$ sur l'évaluation Quaero. La première colonne donne le nombre de titres correctement détectés par l'algorithme sur le nombre total de titres à détecter. La deuxième colonne donne le nombre de détections produites par l'algorithme à tort.

Les résultats sont donnés dans le tableau 3.1. Le nombre de titres détectés est suffisamment important pour déduire que l'algorithme fonctionne, au moins dans une certaine mesure. Le nombre de fausses alarmes est très bas, ce qui prouve que le mécanisme de décision que nous avons proposé remplit efficacement son rôle. La question en suspens concerne les 17% de titres qui n'ont pas été identifiés par l'algorithme. L'écoute manuelle des titres concernés a permis de porter nos soupçons sur le glissement fréquentiel.

Chapter 4

Amélioration de la Méthode Shazam

4.1 Robustesse au Glissement Fréquentiel

Le glissement fréquentiel est une distorsion qui rend le son plus aigü ou plus grave. En termes de spectre, cela consiste en une multiplication des fréquences par un facteur constant κ.

Dans le cadre de la méthode de Wang, nous comprenons donc qu'une paire de points étant encodée par le vecteur $[f_1, f_2, t_2 - f_1]$ va devenir après glissement fréquentiel $[\kappa f_1, \kappa f_2, t_2 - t_1]$. Compte tenu de l'utilisation, lors de la phase de recherche, d'un mécanisme d'index *exact*, il semble clair que la méthode ne peut réussir à identifier un extrait sonore qui a subi un glissement fréquentiel.

Afin de pallier ce problème, nous proposons une modification du modèle d'empreinte de la méthode $Shazam^{03}$.

Le spectrogramme de base utilisé dans la méthode $Shazam^{03}$ est obtenu par des transformées de Fourier. L'empreinte que nous suggérons repose sur un spectrogramme obtenu par la concaténation de transformées à Q constant. Les bins fréquentiels que nous utilisons sont donc espacés logarithmiquement et la résolution fréquentielle de la transformée diminue avec la fréquence. Notons que dans le domaine à Q constant, le glissement fréquentiel n'est plus une multiplication des fréquences mais une translation d'un facteur constant.

Les étapes suivantes sont similaires à celles de la méthode précédemment décrite. Le spectrogramme à Q constant est pavé par des rectangles. Au sein de chaque rectangle, le point d'énergie maximale est mis à '1' et les autres à '0'. Le résultat est un spectrogramme binaire. L'algorithme extrait ensuite toutes les paires de points à '1'. Etant donné que nous sommes dans le domaine à Q constant, nous ne parlons plus de fréquence pour désigner l'ordonnée d'un point mais de bin fréquentiel. Ainsi, pour deux points de coordonnées (t_1, b_1) et (t_2, b_2), l'encodage que nous proposons de la paire de points correspondante est le vecteur

:

$$[\widehat{b_1}; b_2 - b_1; t_2 - t_1]$$

avec $\widehat{b_1} = \left\lfloor \frac{b_1}{6} \right\rfloor$, une version sous-résolue de b_1.

La représentation que nous suggérons a l'avantage de bénéficier des mêmes qualités, en termes de robustesse, que celles de la méthode $Shazam^{03}$ avec en outre une robustesse accrue au glissement fréquentiel. Nous avons effectivement démontré au cours de notre travail que, sur le plan théorique, cette nouvelle représentation est robuste au glissement fréquentiel.

4.2 Réduction de la Complexité de Calcul

Lorsque nous avons présenté la notion de clé d'indexation, nous avons fait une analogie avec l'index d'un livre. Les mots sont référencés avec des pointeurs vers les pages qui les contiennent. De manière similaire, en audio-fingerprint, les clés sont référencées avec des pointeurs vers les titres de référence qui les contiennent. Cependant, on peut observer que dans un livre certains mots ne sont jamais indexés. En fait, seuls les mots les plus pertinents et spécifiques sont sélectionnés. Typiquement, les articles, prépositions et adverbes ne sont jamais indexés. Il est donc légitime de se demander si la situation se transpose au contexte de l'audio-fingerprint. La question qui se pose est alors : "y a-t-il des clés qui sont présentes dans quasiment toutes les références ?". Si c'est le cas, ces clés portent peu d'information et ne font qu'alourdir inutilement le processus de recherche.

Une façon de se rendre compte de la situation consiste à étudier la distribution du nombre de références pointées par clé. Si nous prenons une base de références avec son ensemble de clés extraites, nous pouvons évaluer, pour chaque clé, le nombre de références pointées. Si nous répétons l'opération avec toutes les clés extraites, nous pouvons stocker les valeurs dans un histogramme qui, une fois normalisé, est une estimée de la densité de probabilité de la variable aléatoire correspondante. Un tel histogramme est montré en figure 4.1. Nous pouvons voir sur l'histogramme qu'une proportion importante de clés pointe vers un nombre très important de références. Il est raisonnable de penser que ces clés alourdissent inutilement la phase de recherche. Nous proposons donc une phase optionnelle d'élagage permettant de réduire la complexité du traitement.

Pour chaque clé k extraite de la base de références, nous nommons N_k le nombre de références dans lesquelles la clé k apparaît au moins une fois. N étant le nombre total de références, nous définissons la pertinence d'une clé par :

$$s(k) = \frac{N - N_k}{N} \tag{4.1}$$

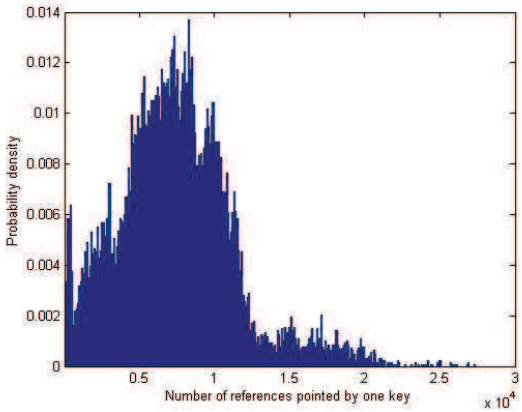


Figure 4.1: Représentation en histogramme du nombre de références pointées par clé. Les clés sont extraites d'une base de 30000 titres. Le bin d'histogramme à l'abscisse x a une ordonnée proportionnelle au nombre de clés pointant vers x différentes références.

En résumé, une clé qui apparaît dans de nombreuses références est peu pertinente. A l'inverse, une clé rare est très pertinente. La phase d'élagage consiste simplement à fixer un seuil T_{prune} et à garder dans l'index uniquement les clés qui vérifient $s(k) < T_{prune}$.

4.3 Expériences et Résultats

4.3.1 Evaluation Comparative

Afin d'évaluer les améliorations par rapport à la méthode originale, nous conservons les conditions expérimentales identiques à celles de la section 3.3. La base contient 7,300 références correspondant à des extraits de 60s de titres musicaux. Le corpus à analyser est constitué de 7 jours de la radio française 'RTL'.

$Quaero_{detec}$	Detected titles / Total	False Alarms
$Shazam^{03}$ [Wan03]	381 / 459 (=83.0%)	0
Improved (CQT-based)	447 / 459 (=97.4%)	0

Table 4.1: Résultats avec notre méthode modifiée sur l'évaluation Quaero. La première colonne montre le nombre de titres correctement détectés dans le flux sur le nombre total de titres à détecter. La deuxième colonne montre le nombre de détections produites par l'algorithme à tort.

Les résultats sont présentés dans le tableau 4.1. Nous pouvons y voir que le taux de détection est beaucoup plus haut avec notre empreinte modifée qu'avec l'originale. Ceci con-

firme que les détections manquées de la méthode originale se sont produites essentiellement en présence de glissement fréquentiel. Ces résultats montrent également qu'en plus d'être robuste aux mêmes distorsions que la méthode originale, notre modèle a une robustesse accrue au glissement fréquentiel.

4.3.2 Evaluation du Passage à l'Echelle

Nous avons mené une deuxième expérience afin de prouver la capacité de la méthode à passer à l'échelle. Le cadre d'évaluation est le même que dans l'évaluation précédente mais nous exécutons maintenant l'algorithme avec une base bien plus fournie. Dans cette expérience, le corpus est constitué de 5 jours de flux radiophoniques provenant de deux stations de radio ('RTL', 'Virgin Radio'). L'ensemble des références est bien plus large puisqu'il contient 30,000 titres.

$Quaero_{filter}$	Detected titles / Total	False Alarms
Improved (CQT-based)	496 / 506 (=98.0%)	0

Table 4.2: Résultats avec notre méthode modifiée sur l'évaluation Quaero. Dans cette expérience, la base de références contient 30000 titres. La stabilité des résultats en dépit du passage à l'échelle montre la capacité de la méthode à gérer des bases industrielles.

Les résultats sont présentés dans le tableau 4.2. Ils montrent que l'algorithme peut passer à l'échelle. La performance de détection est en effet similaire à celle obtenue dans la première expérience alors que la base était plus de 4 fois plus importante. Il est particulièrement remarquable qu'en dépit de l'augmentation de la base, le système a conservé un taux de fausses alarmes nul. Cette expérience montre en conclusion que l'algorithme proposé a la capacité de gérer des bases de données de taille industrielle.

4.3.3 Temps de Calcul

Nous donnons ici quelques chiffres sur le temps de calcul de l'algorithme. Ces chiffres sont obtenus sur la base de notre implémentation en Matlab R 64-bits, exécutée sur un Intel Core 2 Duo @ 3.16 GHz avec 6MB de Cache et 8GB de RAM. Notre implémentation de Shazam a un temps de calcul de 0.08s par seconde de signal. Notre méthode améliorée (à base de CQT) a un temps de calcul de 0.43s par seconde de signal. La différence vient principalement du temps de calcul de la transformée à Q constant. Si nous appliquons la technique d'élagage décrite en section 4.2 avec un seuil $T_{prune} = 0.5$, nous obtenons un gain en vitesse de 35%. Ceci réduit donc le temps de calcul du second algorithme à 0.28s par seconde de signal avec

le même score d'identification. Cela montre que la technique d'élagage proposée permet un gain en complexité significatif tout en gardant des performances similaires.

Chapter 5

Empreinte Basée sur la Transcription en Accords

5.1 Introduction

La méthode présentée précédemment obtient d'excellents scores de détection sur l'évaluation proposée. Cependant, il faut noter que cette évaluation ne contient que des identifications *exactes*. Il s'avère que la méthode ne peut réussir dans le cadre de l'identification *approchée*. Ce n'est en fait pas surprenant étant donné que les caractéristiques extraites du signal sont de relativement bas niveau. En conséquence, on ne peut espérer que ces caractéristiques soient préservées lorsque l'on regarde un autre signal, même s'il existe une similitude musicale. Nous avons donc cherché à proposer de nouveaux modèles d'empreintes qui reposent sur des caractéristiques du signal suffisamment sémantiques pour permettre l'identification *approchée*, autrement dit le rapprochement de deux signaux différents mais musicalement similaires.

La première méthode que nous proposons dans cette optique repose sur la transcription en accords du signal. C'est en fait cette transcription qui servira d'empreinte. Cependant, nous ne considérons pas que cette empreinte est robuste aux distorsions. En conséquence, nous associons un mécanisme de recherche approchée à cette empreinte afin de parcourir très efficacement la base de références.

5.2 Transcription en Accords

Dans le domaine de la transcription automatique, le but ultime serait le développement d'un algorithme qui peut estimer toutes les notes jouées par un ensemble d'instruments.

Ce problème, qui est activement étudié par la communauté scientifique, représente un défi extrêmement complexe et il n'existe pour l'heure pas de méthode fiable pour atteindre cet objectif.

Une façon de contourner le problème consiste à le simplifier. On peut donc se poser la question "n'y a-t-il pas une représentation intermédiaire qui ne nécessite pas l'estimation de toutes les notes jouées mais qui peut cependant apporter de l'information musicale ?". La réponse est "oui" et l'information intermédiaire que nous pouvons cibler est en fait le contenu harmonique. Ce dernier est plus simple à estimer que l'ensemble des notes jouées et il existe de nombreuses méthodes basées sur l'outil chromagramme qui permettent d'estimer la séquence d'accords correspondant le mieux à un extrait musical. Ce qui est satisfaisant avec ces estimations du contenu harmonique est qu'elles ont une réelle correspondance perceptuelle. L'auditeur humain est en effet très sensible à la progression harmonique d'une musique. Le fait d'avoir à disposition des techniques qui estiment l'harmonie présente donc un réel intérêt applicatif, notamment dans notre domaine de l'identification audio.

Plus précisément, nous nous sommes appuyés dans nos travaux sur la méthode de transcription en accords proposée par Oudre **??**. Cette dernière utilise la représentation du signal en chromagramme **??** pour en extraire une séquence d'accords. L'estimation se fait sur la base d'un dictionnaire contenant les accords de référence. Chaque vecteur de chroma du signal est ensuite associé à l'accord de référence le plus proche. La méthode proposée par Oudre prend non seulement en compte, lors de cette recherche du plus proche accord, la distance entre le vecteur de chroma et les accords de référence mais également des statistiques plus globales du signal intégrées au calcul de distance par le biais de l'algorithme EM **??**.

5.3 Distance entre Deux Signaux

Compte tenu de la méthode de transcription en séquence d'accords présentée ci-dessus, nous considérons maintenant que tout signal musical est représenté par une séquence d'accords dans le système. Dans le contexte de l'audio-fingerprint, nous devons mettre en place une méthode pour déterminer, sur la base des transcriptions en accords, si deux signaux sont musicalement similaires. Nous proposons donc une définition de distance dans l'espace des séquences d'accords.

Notre définition de la distance repose sur les considérations suivantes. Si nous avons une correspondance parfaite entre deux séquences, nous considérons naturellement que les deux signaux sont similaires. Mais il peut arriver que deux signaux qui devraient être considérés similaires soient transcris par des séquences légèrement différentes. Il y a deux raisons à cela. La première est que l'estimation des accords n'est pas très robuste. Il arrive notamment assez

fréquemment que le système puisse confondre un accord majeur avec son équivalent mineur. Par ailleurs, l'idée de notre travail sur l'identification *approchée* est d'autoriser un niveau important de distorsion entre deux exécutions d'une même musique. D'une certaine manière, on peut espérer qu'une partie de ces distorsions vont être absorbées dans la modélisation en accords : si deux musiques sont similaires, elles devraient en effet partager la même progression harmonique. Cependant, il peut arriver que d'une version à l'autre, quelques accords soient changés résultant donc en des transcriptions différentes. Par ailleurs, si les deux exécutions ne sont pas exactement au même tempo, il peut arriver qu'une des transcriptions ait quelques accords additionnels (rendant donc le signal plus long).

La nécessité de proposer une distance qui repose sur de la correspondance approchée semble donc claire. La programmation dynamique **??** est une technique qui permet justement de tenir compte des changements qui peuvent intervenir lors de l'estimation d'un accord en particulier mais également de la présence d'accords supplémentaires qui ne sont pas dans la transcription initiale. Nous avons donc mis en œuvre une méthode de programmation dynamique spécifique à notre cas pour le calcul de distance entre deux séquences d'accords.

5.4 Stratégie de Recherche Rapide

Le calcul de distance présenté ci-dessus constitue une base théorique intéressante pour la comparaison de deux signaux représentés par leur séquence d'accords. Cependant, elle est très coûteuse en temps de calcul. En conséquence, lorsque le système doit identifier un extrait inconnu, il serait déraisonnable de calculer sa distance à l'ensemble des références de la base. Un tel système ne pourrait en effet pas passer à l'échelle. Nous avons donc élaboré une stratégie de recherche qui permet de rapidement sélectionner les meilleurs candidats de la base en termes de proximité musicale à l'extrait inconnu.

La méthode de recherche rapide que nous avons retenue est une méthode à indexation utilisée initialement dans la bio-informatique, nommée "Basic Local Alignement Search Tool" **??**. Le principe est de segmenter la transcription inconnue en plusieurs sous-séquences de w accords. Chacune de ces sous-séquences est ensuite utilisée comme requête à un index dans lequel sont référencées toutes les sous-séquences extraites des références. En étudiant la corrélation temporelle des réponses de l'index obtenues pour les différentes sous-séquences avec une référence donnée, la méthode permet de retrouver rapidement les références les plus proches de l'extrait inconnu.

Cette méthode de recherche rapide permet d'accéder rapidement aux meilleurs candidats, cependant, elle n'est pas très précise. Afin d'affiner les performances d'identification, nous utilisons cette recherche 'BLAST' comme une heuristique : elle nous permet de sélectionner

un nombre restreint de meilleurs candidats. Pour chacun de ces candidats, nous calculons ensuite sa distance à l'extrait inconnu par programmation dynamique. Rappelons que cette opération est coûteuse en temps de calcul mais elle est ici limitée à un petit nombre de candidats, ce qui permet de préserver la capacité du système à passer à l'échelle.

5.5 Expériences et Résultats

Nous utilisons le même cadre d'évaluation que dans la section 3.3. Etant donné que les flux constituant le corpus proviennent de vraies stations de radio, ils contiennent occasionnellement des *live* ou des versions acoustiques de certains titres. Si nous incluons dans la base de références les versions studio correspondantes, nous nous retrouvons donc dans un cadre d'identification *approchée*.

Nous avons pu mettre un place un corpus contenant 24h de la radio française 'RTL'. Il contient notamment une émission qui propose essentiellement des versions live de certains morceaux actuels. Au total, le corpus contient 107 titres annotés, dont 99 sont des versions studio et les 8 restant des exécutions en live. La base de références associée à ce corpus est constituée de 2400 titres musicaux.

Les résultats de l'algorithme sont présentés sur une figure ROC classique (voir figure 5.1). Comme la plupart des systèmes de détection, la sortie de notre algorithme repose sur la mise en place d'un seuil de détection. Rappelons en effet que les meilleurs candidats sont sélectionnés lors de l'étape 'BLAST' et qu'ensuite la distance de chacun d'entre eux à l'extrait inconnu est calculée par programmation dynamique. Pour compléter la chaîne de traitement, on peut se contenter de mettre un seuil sur la distance : seuls les candidats plus proches qu'une certaine distance fixée donnent lieu à une sortie de l'algorithme. Dans une telle configuration, on peut évaluer la sortie du système avec différentes valeurs de seuil. C'est tout l'intérêt de la courbe ROC. Chaque point de la courbe correspond aux résultats obtenus par l'algorithme avec un seuil de détection spécifique. Une telle courbe permet donc d'observer la réponse globale de l'algorithme.

Les résultats montrent que la méthode basée sur les accords est capable d'aller plus loin que la méthode traditionnelle (représentée en noir), qui est cantonnée au domaine de l'identification exacte. On peut en effet vérifier que certains points de la courbe ROC rouge ont une coordonnée en Y plus grande que le nombre de titres diffusés en version studio dans le corpus. La contrepartie de cette souplesse accrue dans l'identification est un nombre nettement plus important de fausses alarmes. Ce n'est qu'à moitié surprenant puisqu'en essayant d'assouplir la représentation (afin qu'elle couvre différentes versions d'un même titre) on favorise du même coup le risque de collisions entre deux titres différents.

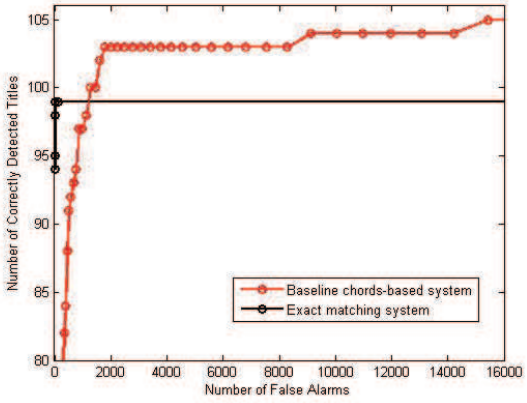


Figure 5.1: Résultats obtenus avec l'algorithme de transcription en accords. La courbe ROC rouge montre les taux de détection obtenus avec différents seuils de détection. La courbe ROC typique d'un algorithme d'identification exacte est représentée en noir.

Chapter 6

Approche Basée sur l'Harmonie et le Rythme

Les résultats ont montré que l'approche précédente a la capacité de dépasser le cadre de l'identification *exacte*. Cependant, cette capacité a été obtenue au prix d'une précision nettement réduite dans l'identification. Le système proposé génère en effet un nombre important de fausses alarmes du fait de la souplesse de sa représentation.

Nous proposons dans cette partie une approche plus complète dont la modélisation met en jeu l'harmonie extraite du signal et également les aspects rythmiques. L'idée est qu'en augmentant la quantité d'information présente dans le modèle, on peut espérer réduire le nombre de fausses alarmes. La difficulté, bien sûr, consiste à ajouter de l'information tout en conservant la capacité du système à faire de l'identification *approchée*.

6.1 Modèle en Etats

Le modèle de signal que nous proposons repose sur l'extraction d'instants musicalement significatifs dans la musique. Dans la suite, nous appelons ces instants $t_1, t_2, ..., t_n$ les *dates d'intérêt*. Dans notre cas, nous déterminons les dates d'intérêt en localisant les pics d'une fonction de détection d'onsets du signal. Notre méthode consiste ensuite à associer à chaque date d'intérêt t_k une information i_k qui caractérise localement le signal. Dans notre travail nous avons choisi, dans la continuité de notre méthode basée sur les accords, de nous appuyer sur une information de type harmonique. Plus précisément, nous stockons dans i_k le vecteur de chroma moyen à gauche de t_k ainsi que le vecteur de chroma moyen à droite de t_k. Nous définissons finalement l'état s_k comme l'association de la date d'intérêt t_k avec l'information locale i_k correspondante :

$$s_k = (t_k, i_k)$$

La représentation du signal est au final donnée par la séquence d'états $\{s_1, s_2, ..., s_n\}$

La figure 6.1 montre la superposition des représentations graphiques d'une fonction d'onset (en blanc) et du chromagramme du même signal Dans notre modèle, les dates d'intérêt sont données par les dates des triangles jaunes sur la fonction d'onset (obtenus par détection de pics). Pour chaque date, le vecteur de chroma moyen à gauche (moyenne des vecteurs de chroma entre le précédent triangle et le triangle courant) et le vecteur de chroma moyen à droite (moyenne des vecteurs des vecteurs de chroma entre le triangle courant et le triangle suivant) sont calculés puis associés à la date, le tout étant stocké dans un état. L'avantage de ce modèle est qu'il est très compact mais conserve de l'information rythmique et harmonique.

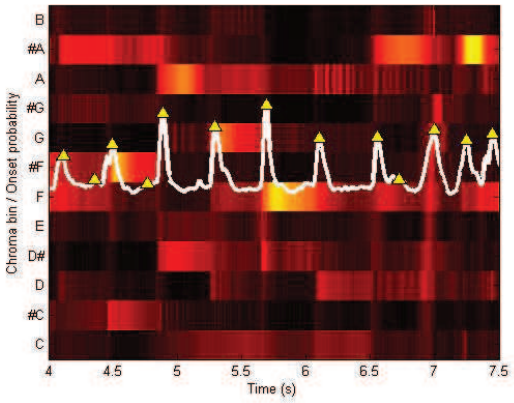


Figure 6.1: Illustration du modèle en états proposé. La courbe blanche représente la fonction de détection d'onsets. Les maxima locaux sont extrait par détection de pics et matérialisés par des triangles jaunes. Le fond est le chromagramme du même signal avec les énergies les plus fortes figurées par les couleurs les plus claires.

6.2 Distance entre Deux Signaux

Maintenant que le modèle de signal a été décrit, il faut définir la distance entre deux signaux, autrement dit la distance entre deux séquences d'états. Pour les mêmes raisons qu'en section 5.3, nous suggérons l'utilisation d'un alignement dynamique. Celui-ci permet de gérer les substitutions d'un état par un autre ainsi que les insertions et les suppressions d'états. Notons que chaque état est caractérisé à la fois par une date et un contenu harmonique. Nous proposons donc un mécanisme d'alignement qui pénalise, lors de la substitution d'un état par un autre, d'une part la synchronisation temporelle entre les deux états et d'autre part la ressemblance entre les harmonies portées par chacun des états.

6.3 Stratégie de Recherche Rapide

De la même manière que pour le système basé sur les transcriptions en accords, il n'est pas concevable de mesurer la distance du signal inconnu à toutes les références de la base. Il faut donc mettre en place une stratégie de recherche rapide permettant de garantir le passage à l'échelle du système.

La stratégie de recherche que nous proposons repose sur la mise en place d'un index. Les clés d'indexation sont extraites de la représentation en séquence d'états présentée. Plus précisément, les clés sont obtenus par une quantification vectorielle sévère de l'information harmonique contenue dans chaque état. Les clés extraites du signal inconnu servent de requête à l'index qui contient toutes les clés extraites des références. Pour chaque référence renvoyée par l'index, nous établissons un diagramme en 2D permettant de visualiser les dates d'occurrence des clés qui sont communes à l'extrait inconnu et à la référence (voir figure 6.2). Pour chaque clé commune, nous positionnons dans le diagramme un point dont l'abscisse est la date d'occurrence dans la référence et l'ordonnée la date d'occurrence de la même clé dans l'extrait inconnu. Lorsqu'il est possible d'observer, dans le nuage de points résultant, une ligne droite, cela signifie que l'extrait inconnu et la référence se correspondent. Les droites en question sont détectées efficacement grâce à la transformée de Hough.

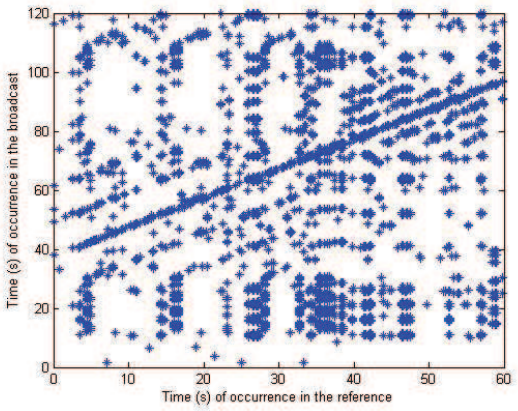


Figure 6.2: Diagramme aggrégeant les sorties de l'index pour une référence donnée. Chaque point de coordonées (x, y) correspond à une clé présente au temps $t = x$ dans la référence et au temps $t = y$ dans l'extrait inconnu.

Cette stratégie de recherche a l'avantage d'être rapide. Cependant, elle est assez peu précise, notamment du fait de la quantification grossière que l'on applique sur le contenu harmonique. De ce fait, il est bon de considérer cette phase de recherche rapide comme une heuristique à la sortie de laquelle nous conservons un petit nombre de meilleurs candidats.

Nous pouvons ensuite calculer la distance de chacun de ces candidats à l'extrait inconnu. Compte tenu du faible nombre de candidats, il est en effet possible de mettre en œuvre le calcul de distance par alignement dynamique, bien qu'il soit coûteux en termes de temps de calcul.

6.4 Expériences et Résultats

Nous menons la même expérience que pour la méthode par transcription en séquence d'accords. Le flux analysé correspond à 24 heures de la radio 'RTL' et contient 8 diffusions qui tiennent de l'identification approchée. La base contient 2400 titres musicaux.

Les résultats sont donnés sous la forme d'une courbe ROC (voir figure 6.3). Pour la comparaison, le graphique présente également la courbe ROC d'un algorithme d'identification exacte (en noir) et celle obtenue avec la méthode par accords (en rouge).

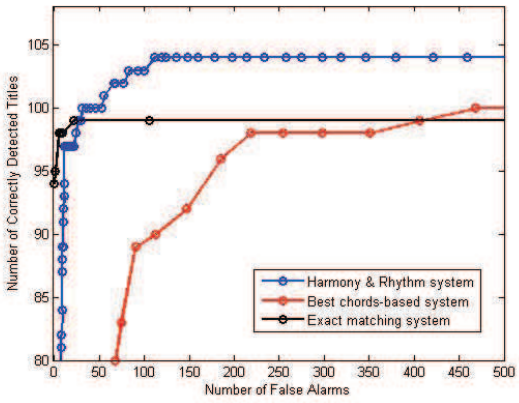


Figure 6.3: Résultats de l'approche basée sur l'harmonie et le rythme sous la forme d'une courbe ROC (en bleu). La coordonnée X donne le nombre de fausses alarmes et la coordonnée Y le nombre de titres correctement détectés. Sont également montrées sur le graphe la courbe ROC d'un système pour l'identification exacte (en noir) et celle du système basé sur la transcription en accords (en rouge).

Les résultats montrent que la méthode, similairement à l'approche en accords, est capable de dépasser le cadre de l'identification exacte. Cependant, nous pouvons observer qu'elle atteint cet objectif avec un nombre bien moins important de fausses alarmes que la méthode en accords. Cela semble montrer qu'en ajoutant une composante rythmique à notre modèle nous avons pu rendre l'empreinte bien plus discriminante, tout en restant robuste au changement de version.

Chapter 7

Conclusion

Au cours de ce travail, nous avons étudié la méthode *Shazam*[03] et avons montré qu'elle présentait une faiblesse vis-à-vis du glissement fréquentiel. Nous avons proposé une modification du modèle qui pallie ce problème et avons observé que les résultats obtenus sur une évaluation extrêmement réaliste devenaient alors excellents. Cette observation, ainsi que les échanges que nous avons pu avoir dans le cadre du projet Quaero, laissent à penser que le cas d'usage traditionnel de l'audio-fingerprint appliqué à la surveillance de diffusion multimédia est un problème aujourd'hui relativement clos. Il reste cependant quelques scénarios d'utilisation permettant de complexifier le problème tels que la détection de musique en fond sonore.

Dans la suite de notre travail, notre objectif a été de proposer de nouveaux modèles d'empreintes qui permettent également de gérer l'identification approchée. Nous avons proposé une approche basée sur la transcription du signal en séquence d'accords qui s'avère avoir une bonne capacité de détection mais génère beaucoup de fausses alarmes. De fait, il serait intéressant d'étudier plus finement les fausses alarmes du système. Celles-ci correspondent en effet à des titres musicaux différents mais qui possèdent des progressions harmoniques similaires. Ce genre de caractéristique peut notamment s'avérer intéressant pour les logiciels de recommandation aux DJs. Notre dernier modèle, basé sur un mélange d'informations rythmiques et harmoniques, semble gagner sur tous les tableaux puisqu'il conserve la capacité à faire de l'identification approchée tout en générant un nombre très réduit de fausses alarmes. Il faut cependant noter que le modèle actuel n'est pas robuste à un certain nombre de variations : en l'état, il ne gère ni la transposition ni le changement de structure.

Part II

Introduction, Context and Use-Case

Chapter 8

General Introduction

8.1 What is Audio-Fingerprinting?

8.1.1 Principles

Content-based audio identification consists of retrieving the meta-data (such as, when dealing with music signals: the title, the artist, the album ...) associated to an audio excerpt. Audio-identification has focused the attention of the scientific community for a decade because of its key role in a number of applications [CBG+02], the two most famous ones being the identification of an audio excerpt through the mobile phone network and the monitoring of copyrighted multimedia content.

There are two popular ways to tackle the problem: watermarking and audio-fingerprint. Watermarking consists of injecting information bits in the audio signal. These bits allow to bring together the expected meta-data. The stake in this approach lies in the injection of the additional information bits that must not alter the audio quality of the signal. Audio-fingerprinting consists of extracting from the raw audio signal, possibly distorted, a fingerprint. The latter allows the identification of the signal that can then be associated with its meta-data. We can note that according to the target application, these two approaches are not necessarily exclusive. Audio-fingerprint can typically constitute a checking step in a processing chain that injects watermarks in audio signals.

Audio-fingerprinting fills the requirements of audio-identification in the following manner. The system has a database of references available (in practice, they are audio files). The references correspond to the set of audio signals that the system will be able to identify. The principle of audio-fingerprinting is to bring the matching of audio signals back to the matching of their fingerprints. Thus, a database of reference fingerprints is compiled from the database of audio references. This is the learning stage (see Figure 8.1). When identifying

an unknown excerpt, its fingerprint is calculated then its best match is looked for in the fingerprints database (see Figure 8.2).

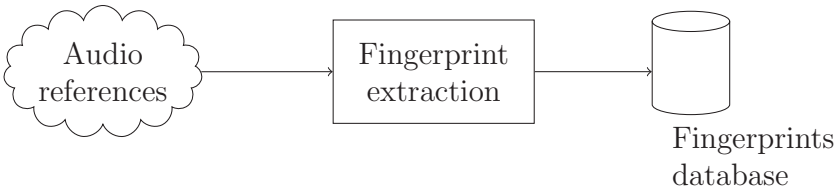


Figure 8.1: Learning of the reference fingerprints by the audio-fingerprint algorithm.

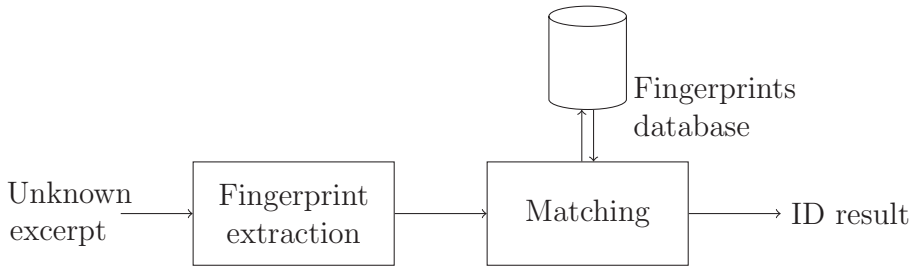


Figure 8.2: Identification of an unknown excerpt by the audio-fingerprint algorithm.

8.1.2 The main stakes

8.1.2.1 Robustness

The fingerprint should ideally allow to identify an audio reference in an unequivocal way. From this point of view, audio identification resembles cryptography, where control functions allow to check in an unequivocal way the integrity of a transmitted code (only the good code generates the good fingerprint). There is however a major difference. Contrarily to cryptography, where the slightest change in the transmitted message must provoke a modification of the fingerprint, audio identification tries to use representations that, in spite of being unique for one reference, are invariant to several modifications of the signal. This objective actually covers two situations that, in spite of their occasional indistinguishability to the human listener, are quite unalike for a computer.

- In the first case, the idea is to identify an audio excerpt that is an exact copy of one reference. We say that both signals are equivalent. The identified signal may however have undergone several distortions that occur in the transmission chain of the signal. Such distortions are named *post-processing distortions* and they are made explicit in section 8.1.3. This configuration, illustrated in Figure 8.3, is referred to as *exact matching*.

Figure 8.3: Illustration of the *exact matching* use-case. The algorithm has to match two signals that correspond to the same recording with different post-processing distortions.

- In the other case, the audio excerpt is not explicitly in the reference database. On the other hand, the database contains one reference which is musically very close to the searched signal. In our experimental context the typical situation is the matching of two different versions of one same title: the database contains the studio version of one song while a live performance of the same song is to be identified. We say that the signals are *similar*. The extreme example is the re-recording of a title as is. Although both recordings will sound very similar to the human listener (it is the same title, performed by the same group of musicians), the little variations that exist between both versions (small rhythm shifts, melodies slightly changed -typically on the starts and ends of the phrases-) make them two radically different signals for a computer. We talk about *approximate matching*, as illustrated in Figure 8.4. Let us note that, in this context, the signal to identify may also have undergone some *post-processing distortions*, as in the first case. The *approximate matching* use-case is thus a strict extension of *exact matching*.

8.1.2.2 Scalability

The other stake in audio-fingerprinting is the scalability of the approach. In the context of industrial applications, the references databases do indeed contain hundreds of thousands of music titles, not to say millions. Typical examples are the databases used by music

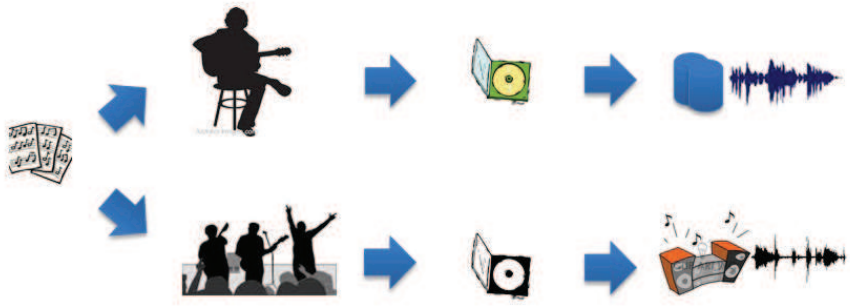


Figure 8.4: Illustration of the *approximate matching* use-case. The algorithm has to match two signals that correspond to the same title performed in different conditions (different arrangement, different musicians, different singer, different recording conditions...).

right management companies, which monitor the commercial channels broadcasts. These contain a number of titles whose order of magnitude is the million. The same holds for the open-source initiatives (Music Brainz, Echo Nest...) whose catalogue entries now reach the million. The ability of an audio-fingerprint system to perform an efficient search is thus one of the fundamental requirements, if not the main requirement. In order to reach such scalability many works propose a fingerprint model that integrates within an indexing scheme. Indexation is indeed a traditional key to the setup of very efficient search models, which is the issue discussed in section 8.1.4.

8.1.3 Traditional Post-Processing Distortions

As mentioned a fingerprint is expected to be robust to the distortions that a signal undergoes in a classic transmission channel. This includes the sound processing that the broadcast station has set up and the distortions induced when the user captures the excerpt. In this section, we detail the major audio processings to which a fingerprint should be robust to. The study is mostly based on the common sound-processing performed by radio stations for the reason that they provide a very representative set of distortions for most use-cases. In Table 8.1, we give the mathematical formulae corresponding to the described distortions. To do so, we consider that the audio signal is represented by a function S of two variables t (for time) and f (for frequency). It is indeed well known that an audio signal can be locally factorized as the weighted sum of harmonic functions with frequencies comprised between 0 and $\frac{f_s}{2}$, f_s being the sampling frequency of the signal. $S(t_0, f_0)$ represents the weight of the harmonic function of frequency f_0 when studying the signal around t_0. L being the length of

the signal, the latter is thus entirely defined by the function:

$$S : \left| \begin{array}{ccc} [0; L] \times [0; f_s/2] & \longrightarrow & \mathbb{C} \\ (t, f) & \longmapsto & S(t, f) \end{array} \right.$$

When dealing with audio identification, many use-cases involve the capture of an excerpt of audio signal. It is seldom the case that the borders of this excerpt coincide with the ones of the signal that has to be matched with. This is why it is usually demanded that a fingerprint system succeeds in spite of the cropping of the signal. Table 8.1 gives the expression of a cropped signal of length M (with $M < L$) starting at t_0 in the original signal. Before broadcasting a signal, all radio stations perform equalisation. It consists of favouring some frequency bands and penalising others. The typical setup for modern music favours the low frequencies in order to emphasise the rhythmic part (drums and bass guitar). In order to limit the dynamic range of a title, radio stations use compressors. This processing is particularly useful when a title is listened to in a noisy environment such as a car. Compressors improve the user experience by raising the gain in low passages. It is common that radio stations apply a pitch-shifting on the titles they broadcast. They generally want the music to sound higher, so that the listener has the impression that the music is more lively. From a signal processing point a view, pitch-shifting is a dilatation of the frequencies. We can note that radio stations perform pitch-shifting thanks to a basic resampling. In this case, pitch-shifting goes together with time-stretching. The time dilatation factor is then the inverse of the frequency dilatation factor ($K = 1/K'$ in Table 8.1). In most use-cases, we have to deal with other effects that are linked with the transmission channel that the signal has gone through. This includes the intrinsic noises linked with the encoding operations, but also noises that can be added when the signal is acquired (background noise). All this can be modelled as the addition of a noise signal to the original signal. In accordance with the classic use-cases, we consider that the added noise is not prevailing over the signal.

There are a number of other processings that can be applied to the signal in classic use-cases. We can think for instance of various sound modules that radio stations use (enhancers, stereo wideners, limiters, ...). Though, their effects are negligible compared with what has been detailed in this section. More details about the usual radio processings can be found in [CBMN02].

Distortion	Formula
Cropping	$\tilde{S} : \begin{array}{rcl} [0;M] \times [0;f_s/2] & \longrightarrow & \mathbb{C} \\ (t,f) & \longmapsto & S(t+t_0,f) \end{array}$
Equalization	$\tilde{S}(t,f) = S(t,f).h(f)$ with $h : [0;f_s/2] \longrightarrow [0;1]$
Compression	$\tilde{S}(t,f) = S(t,f).g(t)$ with $g : [0;L] \longrightarrow [0;1]$
Pitch shifting	$\tilde{S}(t,f) = S(t,Kf)$
Time stretching	$\tilde{S}(t,f) = S(K't,f)$
Addition of noise	$\tilde{S}(t,f) = S(t,f) + n(t,f)$

Table 8.1: Formulae of the post-processing distortions that are handled in traditional audio-fingerprint use-cases.

8.1.4 Indexing

Indexing consists of listing some particular features of a collection of objects. Each of the listed feature is associated with a list of all the objects containing the feature. In the following of the document, we call the listed features the *keys* (or *index keys*) and, to designate the association between the feature and the involved objects, we say that *the key points toward the objects*.

The appearance of indexing can be defined as "*the time when man first began to do something to make information in written records more accessible*" [Wit73]. Proof of such processes has been found up to the second millennium B.C., which makes indexing a very ancient and well-established technique. The topic was notably of significant importance in the libraries, where the aim is to provide the fastest access to the book of interest for the reader. It is indeed easily understandable that an indexing strategy radically changes the search process in a sizable collection of objects. Let us imagine that someone wants to find a precise paper in a journal. He does not know the title but he knows that it talks about 'Markov'. The brute force approach consists of parsing all the articles of the collection, one by one, until the article is found. Conversely, if an index was built, the reader only has to look for 'Markov' in the index. The reader will then have a set of *candidate* articles that all possess the word 'Markov' in their content. Then, only the pointed articles need to be parsed.

For a same collection of objects, there are many ways to build an index. According to the chosen strategy, the efficiency of the search can dramatically change. We can talk of the *quality of the index*. For example, in a book the table of contents constitutes a form of indexing. It is however an index of poor quality: using the table of contents, our reader will not find his article if Markov is not in the title. For that kind of query, glossaries are much

more indicated. These do indeed index all words that are newly introduced, uncommon, or specialised. By exposing this, we also understand that the quality of an index depends on the expected usage of it.

Let us end these general considerations with the observation that the gain of speed in the search step comes at the cost of the construction time of the index as well as the necessity to store the index somewhere. The index construction must take place before the search is performed. We say that it is an *offline* process.

Having exposed the advantages of an index scheme, we understand why it is a very attractive strategy when designing an audio-fingerprinting algorithm. The tremendous gain of search speed that it brings about is indeed of particular interest for us. The difficulty, though, is that the index is an inherently exact scheme. If the reader looks for the word 'Barkov' in the glossary instead of 'Markov', he will never succeed in finding the desired article. This issue must be studied with care, particularly when working with audio features that are highly variable. Throughout the exploration of the different fingerprint models of our work, the central question will thus consistently be: *what can we define as an index key in our model?* The satellite questions that come with are: how do we build our index queries? and how do we process the index outputs?

8.2 My Contributions to the Domain

My PhD work has relied on three cornerstones that are: a careful study of the state of the art, which allowed to understand the stakes and the precise contours of the domain, the set up of a complete and realistic evaluation framework, which reflects the challenges exposed in the works from the state of the art, and the implementation of a baseline algorithm at the level of the state of the art. Based on these, I have further explored the domain under two rather original angles. I have worked on the application of audio-fingerprint methods to the detection of recurrent motives in a stream. Besides, I have got involved in the specific study of approximate matching in the context of audio-fingerprint.

Regarding the evaluation framework, it has been the result of a collaboration with our project partners from Quaero. More precisely, our evaluation framework is based on a use-case that has the meaningful advantage of being representative of real-world applications. For that matter the data that were provided all come from real-world sources and are the ones used in the production database of the industrial partner from Quaero. Namely, this use-case is the "automatic detection of referenced sound items in multimedia broadcasts". Such a use-case involves most of the challenges presented in the state of the art for the audio-fingerprint algorithms. The elaboration of this framework gave rise to a common journal paper detailing the proposed evaluation framework:

- Mathieu Ramona, Sébastien Fenet, Raphael Blouet, Hervé Bredin, Thomas Fillon and Geoffroy Peeters, "A Public Audio Identification Evaluation Framework for Broadcast Monitoring", *Applied Artificial Intelligence: An International Journal*, vol. 26, no. 1-2, pp. 119-136, February 2012.

As far as the baseline algorithm is concerned, I chose, in accordance with the numerous citations of this work in the state of the art, to implement the system described in [Wan03], traditionally referred to as "Shazam's system". A particular attention has been taken in producing a neat code since I have been targeting the management of industrial-sized databases. There was a subsequent tuning phase during which all the parameters of the method had to be optimally determined. When implementing the method described in the paper, it appeared that the final decision step, consisting of a simple threshold, is too naive to handle the false alarms issue with reasonably good scores on real data. For that matter, I have implemented a more complex final step that relies on the splitting of the query in several sub-queries followed by a a fusion of the local decisions taken on each sub-query.

Thanks to the elaborated evaluation framework, I have extensively tested my implementation of [Wan03]. A critical analysis of the results has put the light on a weakness of the

fingerprint. More precisely it appears that it lacks robustness to pitch-shifting when handling real-world data. This has led me to the design of a modified fingerprint and an adaptation of the method accordingly. The much better results obtained on the same evaluation by the method I propose show that the original weakness is overcome. Two papers were dedicated to the description of this modified fingerprint.

- Sébastien Fenet, Yves Grenier, and Gaël Richard, "Une empreinte audio à base de CQT appliquée à la surveillance de flux radiophoniques", in *Proceedings of the Groupe d'Etudes du Traitement du Signal et des Images (GRETSI)*, Bordeaux, France, September 2011.

- Sébastien Fenet, Gaël Richard, and Yves Grenier, "A Scalable Audio Fingerprint Method with Robustness to Pitch-Shifting", in *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, Miami, USA, October 2011, pp. 121-126.

An adjacent problem to the detection of referenced items in multimedia broadcasts is the detection of recurrent motives in a stream. My study of the state of the art did show that although the question had already been approached in some papers, the detection of recurrent motives stays quite unusual in audio-fingerprinting. The issue is indeed usually tackled thanks to correlation methods that are CPU-demanding. I have showed that audio-fingerprinting methods based on indexing of keys can easily be adapted to this use-case. I have taken advantage of this part of my work to demonstrate the general applicability of the search strategy proposed in [Wan03]. To this aim a collaboration with M. Moussallam has been put in place and has led to the development of a new type of fingerprint, utterly different from the first one. This second fingerprint has been integrated in the system and also tested on the recurrent motives use-case. Both topics that are the proposition of an audio-fingerprint based system for the detection of recurrent motives and the demonstration of the general applicability of the designed system are described in the following paper.

- Sébastien Fenet, Manuel Moussallam, Yves Grenier, Gaël Richard, and Laurent Daudet, "A Framework for Fingerprint-Based Detection of Repeating Objects in Multimedia Streams", in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania August 2012, pp. 1464-1468.

Another aspect that seems missing from the state of the art is the study of music similarity in the context of audio-fingerprint. We have indeed noticed that the problem of approximate matching is hardly considered in the works of the domain. In the best case, authors pretend that their methods have the ability to match signals that are musically similar although their testing is only based on exact-matching data. I however believe that there is a real research

interest as well as practical applications in the extension of the audio-fingerprint schemes to the case of approximate matching. Our work in the context of Quaero has notably exhibited that such a system becomes necessary when trying to match automatic annotations with the ones generated by human operators.

I have consequently got involved in the design of new fingerprinting methods that manage approximate matches. The first method that I propose is based on a transcription of the musical signal in a chords sequence. I have started from the work proposed by L. Oudre [OFG11], which allows the generation of the chords sequence. The state of the art of approximate string matching is packed with numerous search methodologies, some of which I have adapted to the context of chords sequence transcription. In the end, the method that I propose uses an adaptation of the Basic Local Alignment Search Tool [AGM$^+$90] that is applied to the chords transcription of the signal. A paper reporting this work will be written very soon.

- *In Preparation:* Sébastien Fenet, Gaël Richard, Yves Grenier, "A Chords-Transcription Based Audio-Fingerprint Method For Approximate Matching".

The first method shows promising results but generates a quite high number of false alarms. The second method that I propose involves a representation of the musical signal that contains a more complete set of information. The method uses a mixture of rhythmic and harmonic information to compute a fingerprint. A specific search strategy has been designed for this combined model. The results achieved with this method seem to show that this technology is a very good compromise between what is achieved by exact-matching methods and what is achieved by our chords-transcription method. The method is the object of a pending patent and of a currently written paper.

- Sébastien Fenet, Yves Grenier, and Gaël Richard, "Génération d'une Signature d'un Signal Audio Musical", FR Patent Pending 1351752.

- *In Preparation:* Sébastien Fenet, Yves Grenier, Gaël Richard, "An Extended Audio-Fingerprint Method with Capabilities for Similar Music Detection".

Chapter 9

The Quaero Project: Use-Case and Evaluation Protocol

9.1 The Quaero Context

This work has been carried out as part of the European project Quaero[1]. More precisely, it belongs to the work-package 11.4 called "Audio Identification and Fingerprinting". We have taken part in this work-package together with three partners.

- Ircam is another research partner whose task, like ours, has been to provide algorithms that meet the use-case of the work-package.
- Yacast is the industrial partner of the work-package. They have proposed the use-case and provided the necessary data.
- IRIT is a research laboratory that is in charge of the annual evaluation. Their task is to manage the evaluation campaign and to provide the evaluation tools.

An evaluation protocol has been collectively defined [RFB+12]. This work has included an overview of the existing techniques used by the authors from the state of the art for the evaluation of their fingerprint methods, a critical analysis of the reported techniques and the setup of a new evaluation framework meant to faithfully reproduce real-world conditions.

Based on the provided data and the defined protocol, an evaluation campaign has been held every year. The evaluation constitutes an opportunity to measure the performance of the developed algorithms within a formal framework and to locate the evaluated methods within the state of the art.

[1] http://quaero.org

9.2 Specifying the Use-Case

The specifications that have been designed in the context of Quaero are based on the "monitoring of multimedia broadcasts" use-case. This use-case indeed constitutes one of the key activities of Yacast, and more generally, of most companies using audio-fingerprint in their businesses. The principle is the setup of a system that continuously monitors some given multimedia streams (TV or radio channels, web...). The system has previously learnt a database of reference audio sounds (typical examples are copyrighted music titles or commercials). The task of the system is then to report any broadcast of any sound from the reference database in the multimedia streams. Let us note that, as stated in section 8.1.3, the broadcast sounds may have undergone some specific sound processing beforehand. Besides, it is worth mentioning that if the broadcast contains sounds from the reference database, it also contains unreferenced bits of stream.

If we denote by $m_1, m_2, ..., m_N$ the references, by $\tilde{m}_1, \tilde{m}_2, ..., \tilde{m}_N$ their broadcast (and distorted) versions and by n the rest of the broadcast (considered as noise for the algorithm), the task can be illustrated as in Figure 9.2.

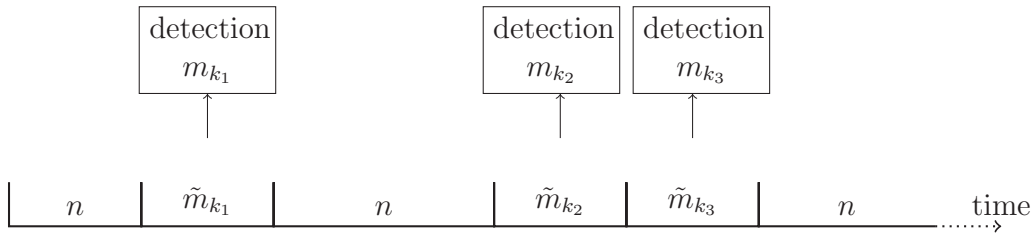


Figure 9.1: Broadcast monitoring use-case. The *automatic annotation system* has to detect in the stream of a multimedia channel all the broadcast sounds from the reference database.

We name a system that handles this use-case an *automatic annotation system*. In our work, we have mostly worked with reference databases comprising only music titles. We consequently refer to the reference sounds as *reference music titles* or *reference titles*. The concepts however stay easily transposable to any kind of sound.

9.3 Evaluation

9.3.1 Traditional Evaluation Techniques

Most of the past contributions in audio-fingerprint use synthetic degradations for the evaluation of the methods. More precisely, the evaluation corpus consists of a subset of the reference database, on which various distortions have been applied. This methodology has

the drawback of skipping a large part of the false alarms issue. As the evaluation set does not contain any item that does not belong to the reference database, this drastically reduces the possibilities of false alarms. In order to overcome this lack, an alternative consists of adding out-of-base items in the evaluation corpus. Another issue with this approach is the difficulty of generating a realistic set of distortions. In order to be representative, the distortions must be combined and tuned in a lot of different configurations, which quickly leads to an untractable experiment protocol. A significant illustration of the weakness of the approach is the lack of experiments including pitch-shifting. Previous authors seem to have considered that it was negligible. Though, our work has shown that this processing plays a crucial role in the performance of some methods on real use-cases.

All these considerations justify the setup of a real-world evaluation. The evaluation corpus then consists of real broadcasts. The advantage of real-world broadcast signals is that they contain a variety of complex combinations among all the presented distortions. Besides, these combined distortions obviously correspond to a real-world use-case. The drawback, in return, is that the level of distortion applied is not controlled.

9.3.2 The Quaero Evaluation Protocol

The evaluation protocol that we set up in the context of Quaero consists of the following steps.

9.3.2.1 Reference database

Before each evaluation the partners collectively decide the size N of the reference database. The references are extracted from an actual production database. For technical easiness, the references are limited to 60s. One reference thus corresponds to 60s of a music title. It has a unique identifier m_i. On the computer, it is one mono 16-bits audio file with a sampling rate of 11.025Hz, whose name is m_i.

9.3.2.2 Corpus

Similarly, the partners agree on a number of days of broadcasts that will be analysed by the algorithms. In order to provide the most representative set of distortions, it is a good practice to use broadcasts coming from different radio channels, preferably with different genres (young, classic, rock, ...). The streams are cut in 5 minutes-long chunks. Each chunk is stored in an audio file whose name contains the string `HHMM`, with `HH:MM` the starting time of the chunk in the day of broadcast. Its end time is naturally `HH:MM+5`.

9.3.2.3 Outputs

Once all the data have been delivered, the participants run their algorithms. In a first step, they have to learn the provided reference database. In a second step, they scan the broadcasts in order to identify the broadcast of items from the database, such as detailed in section 9.2. For each detected item, the evaluated algorithm generates an output that specifies the identifier of the detected item as well as the detection date in the stream.

9.3.2.4 Groundtruth

Meanwhile, Yacast (the industrial partner) has provided the evaluator with groundtruth files, manually generated by professional annotators. These files contain, for each broadcast of a referenced music title, an annotation that stipulates the identifier of the broadcast title as well as a start date and an end date. These dates respectively correspond to the beginning and the end of the whole music title in the stream, as appreciated by the annotator.

9.3.2.5 Scoring

Several scoring methodologies are considered in the article [RFB+12]. The issue is notably discussed in the light of the fact that the reference files only contain 60s of signal (which is generally shorter than the whole music title). For the sake of clarity, we do only consider in this work one type of score. The chosen score fits the recommendations formulated in the article. It is constituted of two components.

The first component is the ratio of successful detections. It gives the number of correctly detected broadcast music titles from the reference database. The titles to detect are the ones listed in the groudtruth files. Each broadcast of a referenced title is annotated with an identifier m_{gt}, a start date d_{gt}^{start} and an end date d_{gt}^{end}.

Definition 3 *One broadcast title is correctly detected if: there exists at least one output whose identifier m_{out} and detection date d_{out} are such that:*

$$\begin{cases} m_{gt} = m_{out} \\ d_{gt}^{start} \leq d_{out} \leq d_{gt}^{end} \end{cases} \tag{9.1}$$

It is a good practise to display the ratio of successful detections under its fractional form:

$$\frac{\text{Number of correctly detected titles}}{\text{Total number of titles to detect}}$$

In this way, the reader gets an instant idea of the size of the corpus.

The second component of the score is the number of false alarms. Each detection output by the algorithm is defined by a reference identifier m_{out} and a detection date d_{out}.

Definition 4 *One detection is a true positive if: there exists one annotation in the groundtruth whose identifier m_{gt}, start date d_{gt}^{start} and end date d_{gt}^{end} are such that:*

$$\begin{cases} m_{gt} = m_{out} \\ d_{gt}^{start} \leq d_{out} \leq d_{gt}^{end} \end{cases} \tag{9.2}$$

All the output detections that are not *true positives* are *false alarms*. The second component of the score is simply the absolute number of false alarms. Given the definition of the use-case (see section 9.2), there is indeed no theoretical limit for the number of false alarms. It thus makes no sense to express this number in a ratio. Besides, as we will see later, the fingerprinting algorithms produce very few false alarms. This also justifies the use of the sole absolute value as a score.

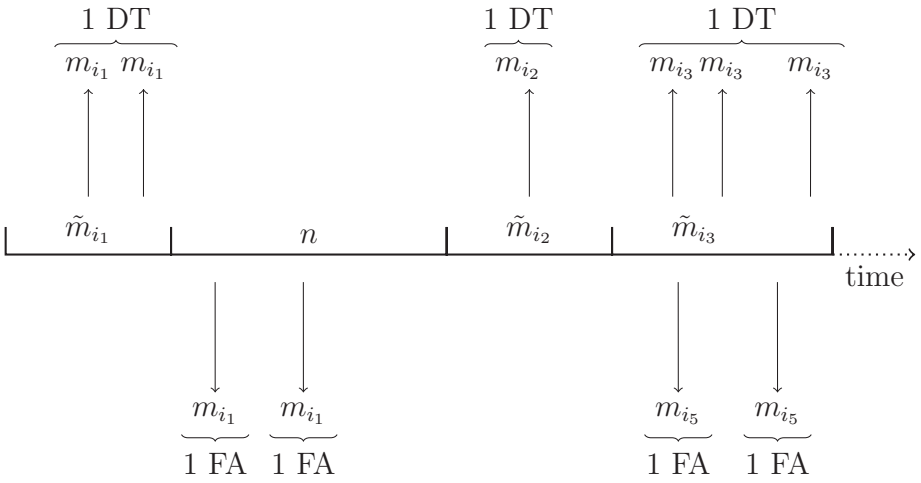


Figure 9.2: Illustration of the scoring procedure. If one output detection (figured by an arrow) contains the identifier m_i and has a detection time that lies between the annotated start time and end time of one occurrence of the title m_i, this broadcast is a *detected title* (DT). Multiple detections of the same occurrence of one title are counted once (as shown by the overbraces). Conversely, if the algorithm detects a reference during an empty slot (denoted by n) or during a slot that contains another title, we count one false alarm (FA). There is no integration on the false alarm counting: each false alarm output is added up.

9.3.2.6 Detection versus tracking use-case

As such, the evaluation protocol corresponds to a pure detection problem. By this, we mean that what is taken into account is only the ability of the evaluated system to output a localised detection during the broadcast of a learnt sound. Indeed, the framework does not consider any form of temporal integration. In particular, this means that the evaluated system is not expected to detect the start and the end of the sound's broadcast. Let us also note that there is no penalisation for a system that would output the same detection several times during one same sound's broadcast. In the following, we refer to this evaluation protocol as $Quaero_{detec}$.

There are several reasons that account for the choice of a pure detection use-case. First, it is noticeable that this detection ability is sufficient for most applications in audio-fingerprint (monitoring of copyrighted contents, identification through mobile network, jingle detection, ...). Second, circumscribing the scope allows to establish clear and accurate metrics whereas assessing different aspects (such as the detection ability and the tracking ability) in a single evaluation makes the definition of the metrics more tricky. Last but not least, our use-case presents the additional difficulty that the boundaries of a broadcast sound are not easily definable. Three different factors account for this situation. The first one is detailed in [RFB+12]. Let us recall that the database only contains 60s excerpts of the music titles. Provided the repetitive structure of most music titles, it is consequently virtually impossible to annotate a precise location of the reference excerpt in the broadcast. The second factor lies in the fact that one title sometimes has several edits (with different intros, codas, featurings, solos, additional bridges...). As a result, even if we had the entire titles available as references, it would still be impossible to state where the broadcast of the reference sound starts and stops, when the broadcast and the reference correspond to two different edits. The third factor is the following: it happens that a radio jingle, or the announcer, overlaps the start and/or the end of the music title. This makes the definition of the start and the end a rather imprecise concept.

After using the protocol $Quaero_{detec}$ on some real-world broadcasts, the necessity of defining a granularity of observation has emerged. The broadcast of music titles on a radio channels is indeed not as rigid as one could think. In addition to the expected broadcast of whole titles, it is indeed frequent that some small bits of music titles are heard. This notably occurs in the following situations: advertisement of a new album, radio games where the listener has to recognise the singer, use of some particular bits of music titles as jingles, broadcast of an excerpt by an animator in order to entertain his guests... Besides, these small broadcasts are frequently overlapped with speech. In order to avoid as much as possible the ambiguous situations, which generate a meaningful cost in terms of annotation and scoring decisions, we

did elaborate a second evaluation protocol whose task is: *"in the analysed stream, detect the referenced items that are broadcast for more than 30s"*. The data, output formats, scoring are kept the same as in $Quaero_{detec}$. The only difference is that there is a filtering on the length of the broadcast reference sounds. We refer to this second protocol as $Quaero_{filter}$.

Part III

Existing techniques

Chapter 10

State of the Art in Audio-Fingerprint

Audio-fingerprint has been a very active field for the last ten years. As a result, the state of the art is packed with a lot of works on the subject. We can notably note that the domain has drawn the attention of industrial actors (Shazam, Philips, Google).

In spite of the diversity of the approaches it is still possible to find a common basis to the diverse audio-fingerprint algorithms that have been proposed [CBKH02]. Let us first recall that any fingerprint system works in a two-step process: it first learns the references database in order to be able, in a second step, to identify unknown excerpts. As for the fingerprint calculation, it can be considered under the following general methodology. In a first step, the signal is pre-processed. This includes a conversion in a standard format (encoding, stereo, sample rate) and a framing with a given length and overlap. This is usually followed by a change of representation. Audio-fingerprint techniques indeed barely rely on the temporal waveform. The latter is too prone to changes when distortions occur. As a result, approaches from the state of the art use representations such as the energy of the signal, the Fast Fourier Transform (FFT), the Discrete Cosine Transform, Mel Frequency Cepstral Coefficients, wavelets... Similarly to a digital fingerprint identification, where the algorithms look for the accidents in the ridges (endings and bifurcations) [JRLJ96], the audio algorithms will then work on extracting characteristic features. There is a double objective: reducing the dimensionality of the representation (in order to make the technique scalable), rendering the representation robust to the distortions. The extracted features are finally gathered in what is called a fingerprint. The latter will be used to uniquely identify the signal.

As indicated before, the main challenge in the design of an audio-fingerprint system is the ability to efficiently browse the reference database when identifying an unknown excerpt. This issue thus often lies at the centre of the proposed algorithms. Through the published works, two types of search can be observed. Authors using the first type consider that

the algorithm is able to extract features that are sufficiently robust to stay unchanged in spite of the distortions. The consequence is that these algorithms can afford the use of exact indexing techniques (hash-tables, B-tree...). Let us note that these algorithms use very local characteristics of the signal (short temporal extent). Indeed, the more the extracted feature is spread in time, the more chances it has to be distorted. The second type of search considers that the extracted features are distorted. These systems thus use search approximate strategies (approximate string matching, exhaustive parsing of the references from the database, retrieval of the K-Nearest Neighbours, ...). In these algorithms, the search step is much more complex than an exact indexing search. To minimise the cost of the search, it is often the case that a preliminary vector quantisation is applied on the features. This, in turn, reduces the search space. These 'approximate search'-based systems use longer characteristic of the signals. Any localised distortion is indeed absorbed in the approximate-search step.

Table 10.1 references the algorithms from the state of the art that we have knowledge of. They are presented along the mentioned axises: model of signal used as a fingerprint, associated search strategy. Besides, the methods are ranked according to their algorithmic similarity. They are notably grouped in four clusters. The methods from the first group extract a fingerprint directly from the temporal representation of the signal. The method from the second group use a Fourier spectrogram as the basis of their fingerprint model. The spectrogram is subsequently filtered by various means in order to end with a compact feature that locally characterises the signal. These methods usually use an index-based search method, that hypothesises that the features are not distorted. The third group of methods extract long-term characteristics from the spectrogram. This can for instance be done by taking the Fourier transform of each energy band of the spectrogram, by using other algebraic projections or by fitting statistical models on the spectrogram. The resulting features are then looked for in the database with an approximate search method. It is noticeable that since we are dealing with long-term features, their rate is rather low. This, in turn, diminishes the complexity of the approximate search. The last group consists of methods that project the signal on a finite codebook (or alphabet). The resulting representation can consequently be seen as a string. If the projection is error-prone, authors will then use approximate string matching methods in order to retrieve the unknown signal among the references.

In the following, we study in detail the fingerprint algorithm proposed by Wang [Wan03]. We subsequently propose a modification of this model but we keep the same search strategy. In consequence, the work presented in this chapter falls within the scope of the second group: extraction of robust features with a small time extent, associated with an exact indexing scheme.

Author	Signal representation	Reduction of the representation	Extraction of keys	Search method
FINGERPRINTS RELYING ON THE TEMPORAL REPRESENTATION OF THE SIGNAL				
Lourens [Lou90]	Energy envelope of the signal.			Exhaustive correlation between the unknown energy envelope and all the reference energy envelopes.
Özer [OSM04]	Estimate of the periodicity of the signal. The trajectory of the periodicity serves as fingerprint.			The work does not deal with the search aspect. A measure of similarity between distorted and original fingerprints is shown.
FINGERPRINTS RELYING ON SHORT-TERM CHARACTERISTICS OF THE SPECTROGRAM				
Pinquier [PAO04]	Fourier transform followed by a piecewise linear filtering resulting in 29 coefficients.			Exhaustive search: Euclidean distance with a sliding windows.
Wang [Wan03] [LcWSI11] [LcWC09]	Fourier spectrogram.	Binarisation of the spectrogram thanks to the detection of local maxima.	Extraction of pairs of local maxima from the binary spectrogram.	Queries to an index with the various pairs from the unknown excerpt. Aggregation of the outputs of the index that takes into account the temporal correlation of the pairs.
Cotton [CE10]	Matching Pursuit decomposition of the signal in an over-complete dictionary.	Keeping of the most energetic atoms.	Extraction of pairs of atoms.	Same as Wang.
Betser [BCR07] [Bet08]	Spectrogram derived from the Discrete Fourier Interpoltor using phase (sinusoidal estimation method).	Asymetric extraction of the most energetic peaks: more peaks are kept in the unknown signal than in the reference (to account for additional peaks that would correspond to additional noise/speech).	Only the frequencies of the peaks are kept (no time, no amplitude).	Look for the frame in the references that has the most peaks (i.e. frequencies) in common. An index strategy for fast search of the peaks is proposed in [Bet08].
Dupraz [DR10]	Fourier spectrogram.	Extraction of peaks.	The frequencies of the extracted peaks are used as keys.	Look for the references having peaks in common within a given tolerance (look-up table). For each returned reference, estimate a pitch-shifting ratio then use it to estimate the temporal correlation between the reference keys and the unknown keys.

Haitsma [HKO01] [HK03]	Fourier transform followed by a Bark scale filtering. An additional autocorrelation step of the Fourier transform is proposed in [HK03] to increase the robustness to pitch-shifting.	Binarisation of the time-frequency representation obtained by a thresholded two-dimensional filtering.	Each temporal frame of the reduced representation is an index key.	Look for the references having in their representation one common key (i.e. one temporal frame) through the index then compare the whole reduced representation of the signals.
Liu [LCY+09]	Similar to Haitsma + a further Discrete Cosine Transform that allows to merge the information of several successive frames in one vector.	Same binarisation as Haitsma on the DCTed representation.	The binarised DCT coefficients are used as keys.	Same as Haitsma.
Schreiber [SGM11]	Same as Haitsma.	Same as Haitsma.	Same as Haitsma.	A prioritisation strategy is proposed so that the algorithm queries the index with the temporal frames that most likely allow a correct targetting of the best reference. Accelerates Haitsma's search.
Ke [KHS05]	Same as Haitsma.	The signal is filtered thanks to binary filters that are learnt out of a wide family of candidates filters beforehand. The candidates notably include the filter used in Haitsma's algorithm.	Each frame of the filtered representation is a key.	The index is queried with the keys. The temporal correlation of the index outputs is evaluated with the RANSAC method.
Baluja & Covell [BC06] [CB07]	Same as Haitsma followed by a wavelets computation on the spectrogram.	The most salient wavelets are kept and the representation is binarised by only keeping the signs of the wavelets.	The information is packed in a reduced signature thanks to the MinHash algorithm.	A Locally Sensitive Hashing performed with the unknown signature returns candidates. The candidates are then compared to the unknown signal thanks to Dynamic Programming.
FINGERPRINTS RELYING ON LONG-TERM CHARACTERISTICS OF THE SPECTROGRAM				
Ramona [RP11] [RP13]	Fourier spectrogram, filtering of the frequencies with a Bark scale, sone-scale amplitude compression (similar to log-compression).	Long-term Fourier transform on each frequency band of the first Fourier Transform. The windows for this Fourier Transform are synchronised with the onsets to reduce the desynchronisation of the frames due to cropping (see distortions in section 8.1.3).	Each long-term frame will be searched for in the database.	K-Nearest Neighbours search among all the (long-term) frames of the references followed by a post-processing that takes into account the temporal correlations of the successive outputs.

Laroche [Lar00]	Fourier spectrogram followed by a log-compression and a filtering of the frequency bands.	Each band undergoes an optional smoothing step and then a finite difference filtering. A long-term Fourier transform on each band is finally computed. In order to avoid the desynchronisation problem due to cropping, the frames of the long-term FFT are set on local maxima of the energy of the signal.	The long-term frames will be searched for in the database.	Euclidean distance with all the fingerprints from the database. The argument of the minimum is the best candidate.
Lin [LOX06]	MFCC.	Windowing of several consecutive MFCCs, modelisation thanks to a Gaussian Mixture Model (only weights are estimated, the other parameters are pre-trained).		Exhaustive comparison accelerated by an 'active search' (derived from time-series theory). It consists of skipping some of the successive comparisons in a stream thanks to a convex bounding of the evolution of the feature vectors.
Burges [BPJ03]	Fourier spectrogram followed by a log-compression and a low-pass filtering.	Oriented Principal Components Analysis applied to the Fourier frames in two layers. In the first projection the top coefficients are retained, then a second projection on a wider time scale is performed. The directions of projections are learnt beforehand by training.		An index-based fast approximate matching is suggested but not detailed. The work presents the results in terms of Euclidean distance between the fingerprints.
FINGERPRINTS RELYING ON VECTOR QUANTISATION OF THE SPECTROGRAM				
Allamanche & Herre [AHH+01] [HAH01]	Fourier spectrogram.	Computation of the Loudness, Spectral Flatness Measure, Spectral Crest Factor.		In a preliminary step, a vector quantisation codebook has been created for each reference. When searching for an unknown signal, it is decoded on each codebook. The codebook with the lowest quantisation error gives the best matching reference.
Cano & Batlle [CBMN02] [BMGC04]	Mel Frequency Cepstrum Coefficients, ΔMFCC and Δ^2MFCC.	Decoding of Hidden Markov Models, called AudioGenes, thanks to the Viterbi Algorithm.	The unknown signal is transcribed into a sequence of AudioGenes, which is equivalent to a string.	Approximate string matching (FASTA) that looks for the closest transcription in the references.
Khemiri [KCPD13] [KPDC12]	MFCC and ΔMFCC.	Decoding of ALISP units that correspond to 3-states HMMs.	Transcription of the unknown signal on the finite codebook consisting of the ALISP units.	Exhaustive approximate string search.

Weinstein [WM07]	MFCC, ΔMFCC and Δ^2MFCC.	Signal cut in pseudo stationary segments, projection on a codebook of Music Phones (Gaussian Mixture Models of the segments).	Decoding of the unknown signal with the Music Phones.	Exact string search with an index scheme in the form of an automaton.
Kurth [Kur02]	Time-frequency representation of the signal.	Binarisation thanks to thresholded finite differences.	The stream is a sequence of 0 and 1, which is transcribed thanks to preliminary learnt codewords. For robustness, codewords were selected with coding theory considerations (error correcting codes).	Index-based search.

Table 10.1: Algorithms from the state of the art in audio-fingerprinting.

Chapter 11

Focus on Shazam's Method

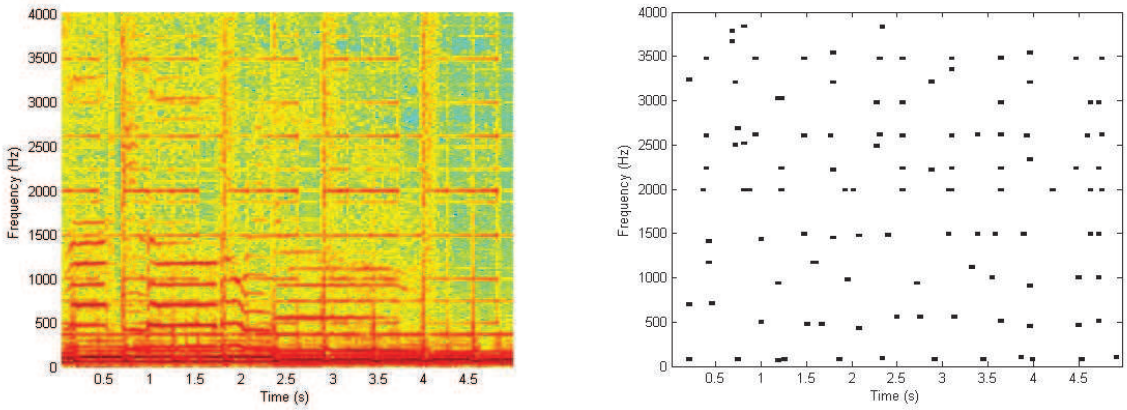
11.1 What is Shazam?

One of the most famous public application of audio-fingerprinting has been developed by Shazam. This company was the first to provide a successful music recognition service on mobile phone. As Shazam's work was the trigger that put audio-fingerprinting in the spotlight, their work is cited in numerous papers of the domain. More precisely, people usually refer to the 2003 ISMIR paper [Wan03] written by Wang. The processing that it describes is commonly referred to as "Wang's method" or "Shazam's method". We should however be more careful than some authors when referring to this work. This paper indeed constitutes, as far as we know, the only conference publication that they wrote about their work in audio-fingerprinting. Though, the interested scientist should note that, aside from this paper, Shazam has registered several patents [LcWC09, LcWSI11]. Besides, as we are dealing with the work of a company, there may always be a difference between what is actually running on their machines and what they release in the public domain through publications. As the large majority of the authors, we focused on the work that is detailed in the article [Wan03]. In the following, we will refer to it as $Shazam^{03}$.

11.2 Principles

11.2.1 Signal representation

As in the vast majority of audio-fingerprinting methods, the representation used in $Shazam^{03}$ is based on a spectrogram of the signal. The raw spectrogram itself is not a robust enough representation for a fingerprint in the sense that it is not invariant to distortions. As in many other works, Wang's idea is to simplify the spectrogram in order to get a lighter representation



(a) (b)

Figure 11.1: Representation of a 5s long audio signal through its Fourier spectrogram (a) and the binary representation suggested by Wang (b) where black points correspond to local maxima in the original spectrogram

that shows little variability. To do so, it is suggested to assign the value '1' to the points of the spectrogram that are local maxima and '0' to all the other points. In this way, the spectrogram becomes a binary representation where active points correspond to local maxima in the original representation (see Figure 11.1). Moreover, it is specified in [Wan03] that the active points must be chosen according to a density criterion.

At this stage, the representation is much lighter than the original spectrogram. The amplitudes have been removed and only local maxima matching the density criterion are still active. In this sense, it is a sparse binary representation. The advantage of this binary representation is that it shows much less variability to the common audio distortions. Indeed, the fact that we do only keep the predominant information makes the representation robust to all distortions that would add little energy to some points of the spectrogram (additive noise) or that would modify the energy distribution without changing the local maxima distribution (equalisation, dynamic compression).

In spite of the binarisation process, the representation is still sensitive to distortions that would add or remove some of the active points. To cope with these distortions, the research step consists of retrieving the database item whose binary representation contains the highest number of common active points with the query. For the sake of scalability, Wang suggests the setup of an index scheme rather than a linear comparison with every item of the database.

11.2.2 Indexing keys

Given the sparse aspect of the binary representation, it seems fairly reasonable to target the active points for the construction of an index. However, Wang's analysis is that using a single point as an index key would not be informative enough. In the binary spectrogram, one point is indeed completely defined by two coordinates: its time location t_0 and its frequency location f_0. As the indexing strategy is expected to be robust to cropping, it would not be possible to use the time information of one single point. This information actually changes when the signal is cropped. This finally means that only the frequency location of the point could be used in the index key. This would lead to indexing queries such as: "Return all the references containing, in their binary representations, at least one active point at frequency f_0". Obviously, this kind of queries is unselective. Virtually all references would pop out at each query, which would imply a meaningful post-processing cost. In order to work around this issue, Wang suggests to work with pairs of points rather than single points. Let us for example consider two active points of coordinates (t_1, f_1) and (t_2, f_2). We can then build an indexing key with three complementary pieces of information. For example, Wang's key consists of f_1, f_2 and $t_2 - t_1$. The strength of this approach is that it allows the use of a time information despite the cropping constraint. This comes from the fact that the time information is now relative. In this way, Wang's keys are much more informative and allow the use of indexing queries that will be less noisy.

This, though, comes at the the cost of the robustness of the key: as it contains more information, it is more likely to change when the signal is distorted. Besides, this approach leads to a higher number of keys in each audio signal: instead of computing N keys, corresponding to the N active points of the binary representation, one should now compute N^2 keys. However, a pruning mechanism that is meant to reduce this computation is included in $Shazam^{03}$.

11.2.3 Research mechanism

When identifying an unknown excerpt, Wang's idea is to find the reference with the highest number of correlated keys. Indeed, if the unknown signal u is an excerpt of reference r_0 starting at time d then all keys appearing in u should be found in r_0. Besides, one key k with time of occurrence $t_{k,u}$ in u should be found in r_0 at time $t_{k,r_0} = t_{k,u} + d$. We thus see that if we study the set of values $\{t_{k,r_0} - t_{k,u}\}$ for all keys k extracted from the unknown signal, we should have a maximum accumulation around value d. So, it is suggested in [Wan03] to store, for each reference r_i, the values $\{t_{k,r_i} - t_{k,u} \,/\, k$ key from the unknown excerpt$\}$ in a histogram

(one histogram per reference). The histogram with the highest maximum corresponds to the best match to the unknown excerpt (see Figure 11.2). We can also deduce that the excerpt starting point in the reference is the argument of the histogram maximum.

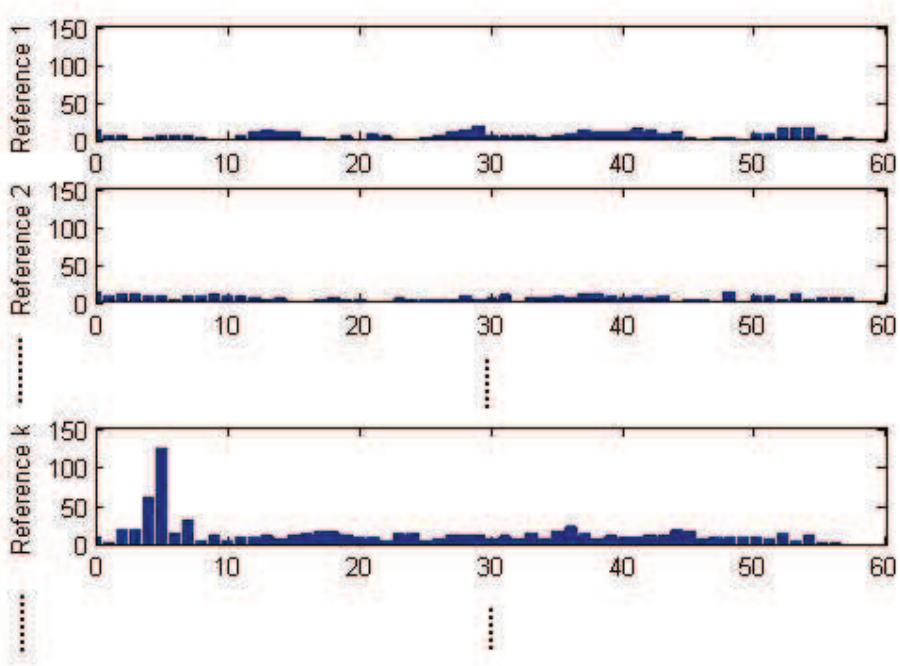


Figure 11.2: Histograms built in the identification phase of an unknown excerpt. Each histogram shows the distribution of the quantities $\{t_{k,r_i} - t_{k,u} \,/\, k$ key from the unknown excerpt$\}$ for one reference of the database. The maximum accumulation gives the best match for the unknown excerpt. Here, the unknown excerpt best matches the excerpt of reference r_k that starts at 5s from the beginning.

11.3 Implementation

11.3.1 Introduction

In [Wan03], the author gives precise guidelines for the processing strategy of his system but very few is said about the implementation details. This makes the analysis of the method tricky since different understandings and different parameters tuning might lead to drastically different results. In this section, we describe our implementation of $Shazam^{03}$ principles as well as our parameters tuning. This section should be particularly useful for anyone wishing to implement $Shazam^{03}$ or any identification system that would work alike.

11.3.2 Binary spectrogram

In order to obtain the spectrogram of the signal, we use discrete Fourier transforms applied on Hamming-windowed frames of length 64ms with a hop t_{hop} of 32ms. The temporal concatenation of the modules of these Fourier transforms gives the spectrogram. The next step consists of extracting peaks that are locally maximum while respecting a density criterion. Let us note that there is no indication on the way to proceed in [Wan03]. We suggest the following methodology: we tile the spectrogram with rectangles of width ΔT_{tile} seconds and height ΔF_{tile} Hertz (typical values are $\Delta T_{tile} = 0.4$s, $\Delta F_{tile} = 400$Hz). In each rectangle, we set the maximum point to 1 and all the other points to 0. This processing, that is simple and quickly computable, ensures the homogeneous peaks density that it is required in $Shazam^{03}$. The adjustments of ΔT_{tile} and ΔF_{tile} allow to independently control the density of peaks on the time axis and on the frequency axis.

11.3.3 Extracting and encoding pairs of peaks

Theoretically, the system should then compute all possible pairs of peaks in the signal. The extracted pairs of peaks must then be encoded. For two given peaks with coordinates (t_1, f_1) and (t_2, f_2) Wang suggests the following encoding: $[f_1, f_2, t_2 - t_1]$. However, in order to prevent an explosion of the number of pairs, a pruning technique is suggested in [Wan03]. In concrete terms, we only consider pairs of peaks whose spectral extent $f_2 - f_1$ is smaller than a threshold ΔF_{max} and whose temporal extent $t_2 - t_1$ is smaller than a threshold ΔT_{max} (typical setup for this limitation is $\Delta T_{max} = 3$s and $\Delta F_{max} = 350$Hz).

In his paper, Wang gives an indication at a surprisingly low level of programming, which is that the information $[f_1, f_2, t_2 - t_1]$ can be packed on 32 bits. In practice this comes to allocating n_1 bits for the encoding of f_1, n_2 bits for f_2 and n_3 bits for Δt with n_1 n_2 and n_3 such that $n_1 + n_2 + n_3 = 32$. When encoding a variable on a given number of bits, one should cross multiply its value so as to bring its original span to the span allocated by the number of bits. In this way, the code for f_1 is given by:

$$\widetilde{f_1} = \left\lfloor \frac{f_1}{f_{max}} . (2^{n_1} - 1) \right\rfloor$$

with f_{max} the maximum frequency of the spectrogram. The same formula goes for $\widetilde{f_2}$ with n_2.

Due to the pruning technique applied we know that Δt is bounded by ΔT_{max}. This

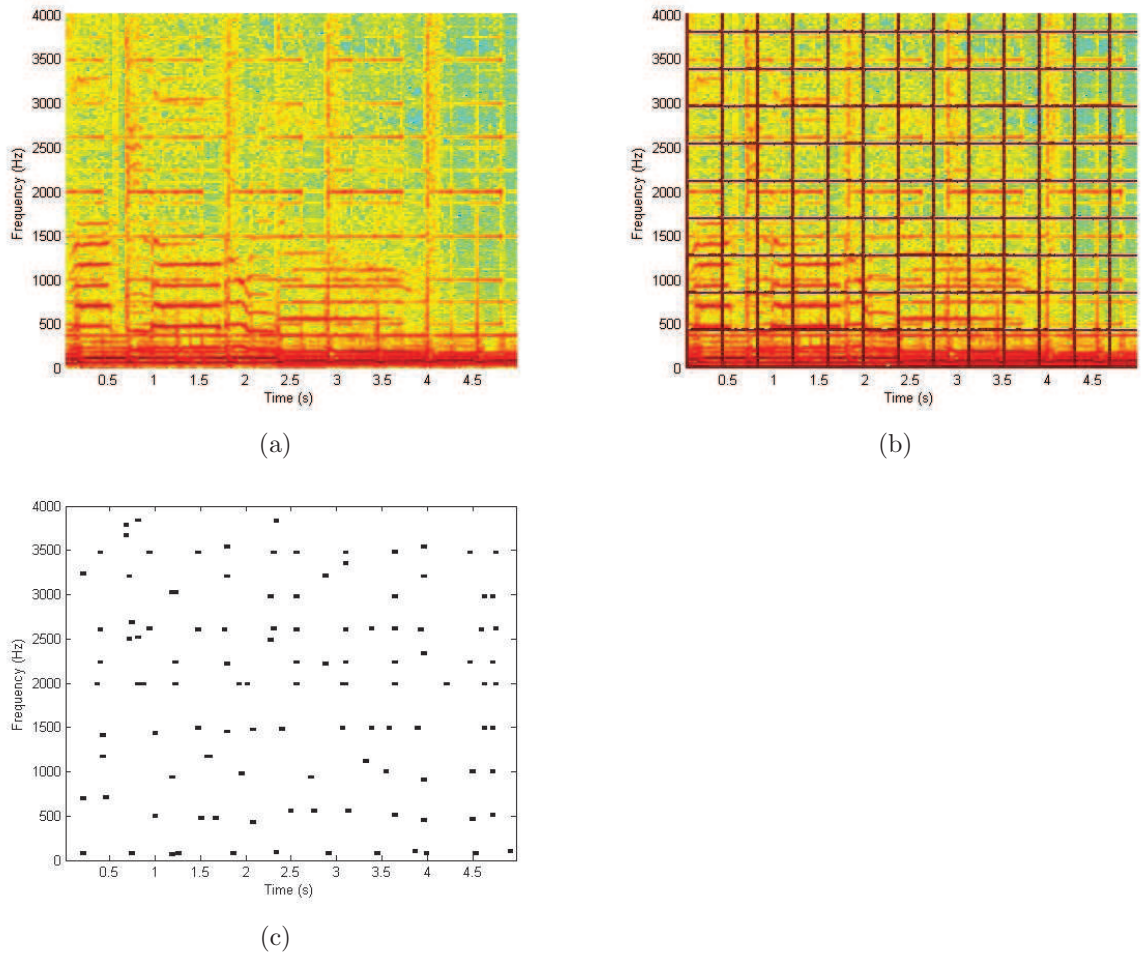


Figure 11.3: Computation of the binary spectrogram (c) proposed in *Shazam*[03] thanks to a tiling (b) applied on the spectrogram of the signal (a). In each tile, the maximum point is set to 1 and the others to 0. This results in the binary spectrogram with points set to 1 in black and points set to 0 in white.

guarantees that the following formula will not cause any overflow:

$$\widetilde{\Delta t} = \left\lfloor \frac{\Delta t}{\Delta T_{max}}.(2^{n_3} - 1) \right\rfloor$$

Although this issue is not mentioned in Wang's paper, we should take advantage of this formula to carefully think about resolution. Indeed, in the preceding formula, the encoding resolution of Δt is given by $\frac{\Delta T_{max}}{(2^{n_3}-1)}$. However, we have a theoretic lower bound for this resolution. Δt is actually extracted from the spectrogram. The resolution of the temporal indices of the spectrogram is t_{hop}. As a consequence, the resolution of Δt is $2t_{hop}$ (difference of two time indices with resolution t_{hop}). For this reason, Δt should never be encoded with a finer precision. Otherwise, we would be encoding computation artifacts which may change when analysing other recordings of the same signal (due to the possible cropping, the Fourier transforms will not be synchronized exactly the same way ...). As a conclusion, the following inequality should always be verified:

$$\frac{\Delta T_{max}}{(2^{n_3} - 1)} < 2t_{hop}$$

The numerical representation of the key is finally obtained by concatenating the three components. The concatenation in the binary domain is given by:

$$\widetilde{k} \;=\; \widetilde{f_1} \times 2^{n_2+n_3} \;+\; \widetilde{f_2} \times 2^{n_3} \;+\; \widetilde{\Delta t}$$

The binary representation of this key $\widetilde{k}$ fits inside 32 bits. The first n_1 bits are the binary representation of f_1, the next n_2 the binary representation of f_2 and the last n_1 bits the binary representation of Δt.

11.3.4 Storing the references keys in the database

The learning stage suggested by Wang is the following. For each reference, all its keys are extracted and encoded in their 32-bits versions. Each key is then associated to its value and recorded in the index engine of the system. For one given key, the associated value contains two pieces of information: the time of occurrence t_1 of the key in the audio reference, an identifier of the reference (in concrete terms, it suffices to number the audio references and to consider this number as an identifier). Wang recommends to encode these pieces of information on 32 bits as well (same process as for the binary key $\widetilde{k}$). In the end, we thus have an index engine that must manage queries consisting of 32 bits keys and that returns 32 bits values. Let us note that one key can be found in several references at several dates,

and that for this reason, the index engine must be able to return several values when it is queried with one key. Conversely, it might happen that the index is queried with one key that does not exist in the references, in which case the engine should return an empty set. As the amount of data that will be stored is quite substantial, the choice of the index engine is quite crucial. In the course of our work, we have tried several solutions, most of which have appeared to be unsuitable for a scalable system.

- The first solution consists of storing all the keys in a table of the Random Access Memory. The first difficulty is that each key can be associated with several values. This compromises the use of a static table. The setup of a dynamic table can be done with two tables. The first table (let us name it K_{table}) contains the keys and the second (let us name it V_{table}) the values. Each stored pair key/value corresponds to an index i. So, $K[i]$ contains the key of the pair, whereas $V[i]$ contains the value. Given that a key can be associated to several values, we possibly have $K[i] = K[j]$. In order to perform the searches in a reasonable time, the keys table must be sorted so that a binary search can be performed (thus leading to a $O\log(n)$ complexity). In the end, the search complexity is reasonable but the construction of the tables is quite challenging. This occurs because the number of keys is not known in advance and the table must be sorted. Let us indeed recall that inserting an element in the middle of a table, in terms of memory accesses, corresponds to rewriting the entire table. When dealing with extremely large datasets, this can lead to untractable construction times.

- In order to get rid of the dynamic approach (that costs $O\log(n)$ in the search complexity) we have tried to implement a static approach. That consisted of allocating large memory blocks for each key so that all values associated to this key could be written in the block. This, in turn, led to a large memory requirement for the structure. It had to be stored (and accessed) on the hard-disk. The conclusion was that the introduction of Hard-Disk accesses was largely counterbalancing the gain of the binary search, leading to an unacceptable algorithm in terms of search time.

- Given the complexity and the specificity of the use-case, we have taken the option to look into the established database engines. We were looking for an engine that works stand-alone (it is preferable to avoid the client-server problematic as it does not bring any advantage in our use-case). Our testing tended to show that relational database engines (such as SQLite [Hip13]) were not fast enough for our use-case. Let us remember that our system queries the database with a numerous number of 32-bits keys per second and expects the associated 32-bits values in return (exact query). The SQL-like engines are designed to process more complex queries, with a large diversity

of types in the entries but are not particularly fast when processing exact queries. The solution consequently seem to lie in the field of hash-tables. We have tried Tiny CDB [Tok12], but its memory limitations can be disturbing when going to industrial-sized databases of audio references. Finally, Berkeley DB [Ora12], set to its Hash-Table mode, provided us an efficient way to deal with our problem. Our Matlab programs could be interfaced with BDB thanks to the C API that is proposed. Our initial experiments showed that this engine was satisfyingly meeting our needs. Let us note that in the entire rest of our work, we have kept working with BDB.

11.3.5 Merging the database outputs

In the identification phase of $Shazam^{03}$, all pairs of peaks are extracted from the unknown excerpt according to the methodology described above. These are then converted into database keys which are used to query the database engine. As detailed in section 11.2.3, the outputs of the database engine must be stored in histograms. Practically, if, for a key k occurring at t_u in the unknown excerpt, the database engine outputs the value v_i, it is decoded into its two components: the identifier of the reference r_i possessing the key k and the time of occurrence t_{r_i} of the key in this reference. Then, $t_{r_i} - t_u$ is stored in the histogram corresponding to r_i. In order to build the histograms, one should choose a time resolution δt. Knowing that the reference songs are L_{ref} seconds long, this leaves us with histograms containing $L_{ref}/\delta t$ bins. Finally, the entire set of histograms can be represented in a static table of size $N \times L_{ref}/\delta t$. Each output of the database engine thus gives rise to an increment in the histogram table. Let us note that this operation is repeated a meaningful number of times (a typical rate is 250 per second of signal). For some reason, this incremental operation (that takes place in a large table) is quite slow in Matlab. Consequently, we have recoded this part of the code in C. It has been interfaced with the rest through the Mex API. Finally, in order to perform the identification we look for the reference whose histogram has the highest maximum. This reference is considered to match the unknown signal. The argument of the maximum of the histogram gives the start time of the unknown signal in the reference.

11.3.6 Fusion of local decisions

Whatever the query, the previous step returns the best match with its start time. This means that the case of a query that does not correspond to any excerpt in the database (out-of-base query) is not managed.

As far as this issue is concerned in Wang's paper, it is suggested to setup a threshold mechanism. More precisely, the threshold is set on the number of correlated pairs between

the unknown excerpt and its best match (which corresponds to the height of the maximum histogram peak). If the unknown excerpt has more than *threshold* pairs correlated with the best match, the identification is correct. Otherwise it is an out-of-base query. However, our experiments have proved that such a threshold is virtually impossible to setup on real data with classical distortions. It happens that, due to the distortions, a best match has a low number of pairs in common with the unknown signal even though it is a correct identification. In terms of probability density functions: the distribution of the False Alarms has a large overlap with the distribution of the True Positives in the domain of the correlated number of pairs. Besides, such a *threshold* would depend on the transmission channel and would have to be tuned for each different use case.

This is why we propose our own post-processing method. Let us note that this constitutes a true theoretical contribution to the method $Shazam^{03}$. The method we propose is based on the fusion of local decisions. The idea is to give a multi-scale aspect to the problem in order to add robustness to the decision. The unknown signal u is split into sub-signals u_i^{sub} with a given length L^{sub} and overlap o^{sub}. The post-processing considers P successive sub-signals $\{u_j^{sub}\}_{j=1..P}$. Each of them has gone through the identification procedure described above, resulting in a matching result (r_i, s_i) consisting of the best candidate identifier and the excerpt's start time in the best candidate. If among these P identifications, more than T_{vote} of them are coherent the best match is considered to be a correct identification. Otherwise, it is an out-of-base query. Two matching results (r_i, s_i) and (r_j, s_j) of the i^{th} and the j^{th} sub-signals are coherent if:

$$\begin{cases} r_i = r_j \\ s_i - i.L^{sub}.(1 - o^{sub}) = s_j - j.L^{sub}.(1 - o^{sub}) \end{cases} \tag{11.1}$$

The tuning of T_{vote} determines the system's sensitivity. T_{vote} can take any integer value between 0 and P. If $T_{vote} = P$, we require that all matching results of the sub-signals are coherent. In this way, we minimise the risk of False Alarm. The counterpart is the risk of missing some good detections. Conversely, if T_{vote} is set very low, we generate a lot of False Alarms but we maximise the good detections. In practice, a reasonable value for T_{vote} is:

$$T_{vote} = \left\lceil \frac{P}{2} \right\rceil \tag{11.2}$$

Name	Description	Value
L_{FFT}	Length of the FFT windows	64ms
t_{hop}	Hop between the FFT windows used to compute the spectrogram	32ms
L_{sub}	Length of the sub-queries	5s
o_{sub}	Overlap between the successive sub-queries	0.5
ΔT_{tile}	Width of the rectangles used for the tiling of the spectrogram	0.4s
ΔF_{tile}	Height of the rectangles used for the tiling of the spectrogram	400Hz
ΔT_{max}	Maximum time extent between two points of the binary spectrogram to be paired together	3s
ΔF_{max}	Maximum frequency extent between two points of the binary spectrogram to be paired together	350Hz
n_1	Number of bits used to encode f_1	13
n_2	Number of bits used to encode f_2	13
n_3	Number of bits used to encode $t_2 - t_1$	6
δt	Time resolution of the histograms	1s
P	Number of sub-queries used in the fusion of local decision (post-processing)	6
T_{vote}	Threshold above which one best match is considered as a correct detection	3

Table 11.1: Summary of the parameters used in our implementation of $Shazam^{03}$

11.4 Experiments and Results

11.4.1 Adapting the system to continuous broadcasts

As indicated in section 9.2, our work was mostly focused on continuous broadcasts. For this reason, the audio-fingerprinting techniques that we test are always preceded by a framing module. The latter cuts the input broadcast in frames of length L_a with an overlap rate o_a. These frames are called *analysis frames*. For practical reasons, we take $L_a = L^{sub}$ and $o_a = o^{sub}$. This means that the system performs a best match search on every analysis frame. It then has to post-process a window of P consecutive analysis frames to make a detection decision. The hop of the post-processing window is set to 1 analysis frame.

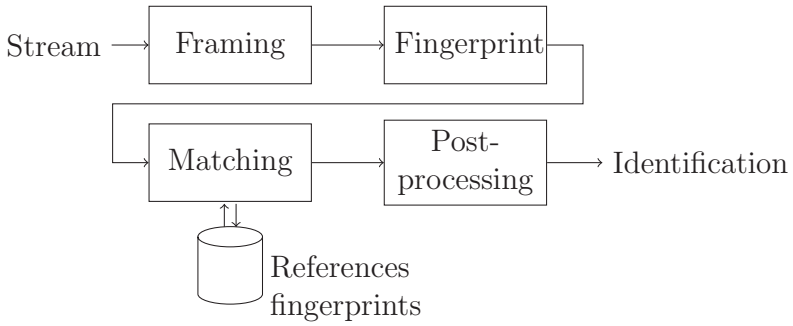


Figure 11.4: Architecture of the system. The traditional blocks of a fingerprint system are preceded by a framing module which partitions the input stream in analysis frames of length L_a with an overlap o_a.

11.4.2 Proof of Concept

Before moving to the challenging real-world evaluation described in section 9.3.2, it is worth testing the approach on a simpler task. This should allow us to determine if the designed fingerprint is discriminative enough. This can also help checking that the code does not contain any obvious bug. We consequently design a first evaluation step that we call *Proof of Concept* (PoC) in the following way.

The idea is to build an artificial stream by concatenating all the reference titles from the database. Each reference m_i has a length l_i. We define n_i by

$$n_i = \sup\{n / nL_a \leq l_i\}$$

We then define $\widetilde{m_i}$ as a truncated version of m_i of length $n_i L_a$. Finally, our artificial broadcast is given by the concatenation of the truncated references: $\widetilde{m_1}\widetilde{m_2}...\widetilde{m_N}$.

In this experiment, the algorithm is in the following configuration: no overlap between the successive analysis frames ($o_a = 0$) and no post-processing step (which is equivalent to setting the parameters P and T_{vote} to 1). This means that the algorithm will process each analysis frame independently. For each of them, it will output the best match from the reference database (since there is no post-processing, there is no out-of-base output). Besides, given the construction process of the artificial stream, each analysis frame is an excerpt of one *and only one* reference of the database. The scoring is straightforward. If the output corresponds to the reference from which the analysis frame originates, it is counted as a 'true positive'. Otherwise, it is a 'false alarm'. Let us note that this experiment does not include any kind of distortion. The references are matched against themselves as is. The PoC essentially measures the discriminative power of the fingerprint.

PoC	True Positives	False Alarms
$Shazam^{03}$ [Wan03]	99.9%	0.1%

Table 11.2: Results of the $Shazam^{03}$ algorithm in the PoC experiment. The algorithm has to identify frames that are directly taken from the reference database. For each frame (taken from one reference), if the best match computed by the algorithm corresponds to the reference, it is counted as a True Positive. Otherwise, it is a False Alarm.

The results are given in Table 11.2. Let us note that the score obtained in the PoC experiment does not constitute an upper bound for the real-world experiment. The PoC is indeed scored on an instant frame-by-frame basis. The algorithm has to take one decision per analysis frame and it is immediately counted as a true positive or a false alarm. In the real world experiment, when a referenced title is broadcast, the algorithm has the whole duration of the song available to make a decision. This notably explains the low number of false alarms that is commonly achieved by the methods of the state of the art.

11.4.3 Real-world evaluation

Since the PoC gives satisfying results, we can move to the real-world evaluation, such as described in section 9.3.2. Let us briefly recall that this evaluation consists of analysing real-world broadcasts coming from radio channels. The algorithm has to detect the broadcast of any music title that belongs to its reference database. In this experiment, the database is composed of 7300 excerpts of 60s of music titles and the analysed stream corresponds to 7 days of broadcast of the French radio 'RTL'. Since the references from the database have not been captured on the same channel as the analysed stream, they are not post-processed in the same way. This ensures a corpus that possesses a meaningful level of distortion. As the dataset used in this experiment is the same as the one of the 2010 Quaero evaluation, we are able to display the results of IRCAM's algorithm [RP11] on this task.

$Quaero_{detec}$	Detected titles / Total	False Alarms
IRCAM [RP11]	445 / 459 (=96.9%)	2
$Shazam^{03}$ [Wan03]	381 / 459 (=83.0%)	0

Table 11.3: Results of our implementation of $Shazam^{03}$ and of IRCAM's algorithm on the Quaero evaluation. The first column shows the number of broadcasts of reference titles detected by the algorithm over the total number to detect. The second column shows the number of wrongly output detections.

The results are given in Table 11.3. The number of detected music titles is sufficiently

high to deduce that the algorithm works, at least to some extent. The number of false alarms is very low, thus proving that the proposed decision mechanism, based on the fusion of local decisions, fills its role very efficiently. The remaining question concerns the 17% of music titles that have not been identified by the algorithm. When zooming on these portions of broadcasts, one can note that the correct music title is hardly output as a best match. The issue is thus independent of the fusion module and lies in the fingerprint model. When manually listening to the concerned music titles, the careful operator can notice that there is a high pitch-shifting ratio between the database version and the broadcast version.

11.5 Limitations of the Method

Our experiments suggest that the method does not work well when dealing with signals that are highly pitch-shifted.

Let us recall that pitch-shifting is a distortion that makes the sound higher or lower [Ber99]. Spectrally speaking, it consists of multiplying all the frequencies of the spectrum by a constant factor κ. It is also noticeable that pitch-shifting is often obtained simply by reading the music faster or slower (i.e. using an output sampling rate that is different from the input one). In this case, pitch-shifting goes together with a time-stretching effect of factor $\frac{1}{\kappa}$. We then talk about *sample rate conversion*.

If we consider one reference of the database, its fingerprint is calculated thanks to the extraction of peaks in its spectrogram (see section 11.2). Considering the above presented distortions, one can see that a point with coordinates (t_1, f_1) will undergo the following transformation:

$$(t_1, f_1) \longmapsto (\frac{1}{\kappa}t_1, \kappa f_1)$$

Consequently, when encoding a pair of spectral peaks, we have:

$$[f_1, f_2, t_2 - t_1] \longmapsto [\kappa f_1, \kappa f_2, \frac{1}{\kappa}(t_2 - t_1)]$$

We should however note that the time-stretching effect is absorbed by the encoding resolution of $t_2 - t_1$. The latter is indeed encoded with a resolution that is not finer that $2t_{hop}$ (see section 11.3). With typical values such as the ones indicated in Table 11.1, this gives a resolution of 0,064s. Knowing that $t_2 - t_1$ is bounded by ΔT_{max}, a time-stretching ratio of 5% induces a variation of $t_2 - t_1$ that is, at maximum, 0,05s. Since $0,05 < 0,064$ the variation will be masked in most cases.

Conversely, we can note that the frequency resolution is given by $1/L_{FFT}$, which corresponds to 15Hz if we use the typical values of Table 11.1. Taking into account a 5%

pitch-shifting, we see that for any frequency above 300Hz the shift is larger than 15Hz. The pitch-shifting effect is thus non negligible with such a frequency resolution.

The distorted pair consequently has, in most cases, two components out of three that have been modified. Since we use an exact index scheme, there is no way that when querying the index with the distorted pair we obtain the value assigned to the original pair. When processing an analysis frame of the distorted signal, the method will thus aggregate database outputs that do not correspond to the original reference. The correct reference will not be output as a best match. Finally, the fusion of the local decisions made on each analysis frame will fail to output the original reference. That explains why the method fails when the signals has undergone a sample rate conversion.

Part IV

Exact Matching

Chapter 12

Improving Shazam's Method

12.1 Addition of a Tracking Step

When targeting applications with a wider functional perimeter than the sole identification use-case, such as synchronised display of the lyrics, automatic segmentation or automatic annotation of multimedia broadcasts, it becomes necessary to develop audio-fingerprint methods that have the ability to track a reference throughout its broadcast. Besides, this tracking requirement matches our evaluation protocol $Quaero_{filter}$ (see section 9.3.2), according to which the algorithm has to control that any detected reference is broadcast for more than 30s. It is interesting to note that the fusion of local decisions module presented in 11.3 already constitutes a kind of tracking. In this context, the detection decision is only taken if one reference appears several times in the horizon of P frames. This can thus be seen as a tracking. However, this tracking has an observation window limited to P analysis frames (typically corresponding to 15s of signal). Hence the necessity to add a further tracking step, managing a wider portion of signal.

The fusion module takes into account P successive analysis frames. These constitute a vote horizon v_i. If one reference appears predominantly as a best match in the vote horizon, the fusion module outputs the detection of this reference. We can add two other pieces of information: the starting time $t_0(v_i)$ of the first analysis frame of the vote horizon having the detected reference as a best match, the end time $t_f(v_i)$ of the last analysis frame of the vote horizon having the detected reference as a best match. In terms of tracking, we know that the detected reference is broadcast at least during the time interval $[t_0(v_i); t_f(v_i)]$.

Let us call $f(v_i)$ the detected reference on the vote horizon v_i (note that $f(v_i)$ can be the empty set if there is no detected reference). When a music title m_0 is broadcast, the following structure of outputs is usually observed.

$$
\begin{aligned}
f(v_i) &= m_0 \\
f(v_{i+1}) &= \emptyset \\
f(v_{i+2}) &= m_0 \\
&\vdots \\
f(v_{i+k}) &= m_0 \\
f(v_{i+k+1}) &= \emptyset \\
&\vdots \\
f(v_{i+k'}) &= \emptyset \\
f(v_{i+k'+1}) &= m_0 \\
&\vdots \\
f(v_{i+k''}) &= m_0 \\
f(v_{i+k''+1}) &= \emptyset \\
&\vdots
\end{aligned}
$$

Several factors account for this structure. First, it might happen that the signal is "corrupted" during some vote horizons. For example, when the announcer speaks on the introduction of the music title. Such a situation is illustrated by $f(v_{i+1}) = \emptyset$. Second, we have to keep in mind that the reference only consists of 60s of the music title. Since music titles possess a very repetitive structure, it is likely that these 60s or at least part of them will be repeated across the music title. That explains the detection of the reference between v_i and v_{i+k}, then the absence of detection and finally the repetition of the 60s reference between $v_{i+k'+1}$ and $v_{i+k''}$. Last, it happens that one music title is edited with different lengths. In concrete terms, some edits possess additional bridges, intros or codas. These additional pieces cause the presence of empty detections during the broadcast of the music title.

In order to take into account this output structure, we suggest the following tracking mechanism. When a reference (m_0) is firstly output (at horizon v_i), the tracking module looks for the last m_0 output within the M following horizons ($v_{i+k''}$). M is chosen sufficiently long to contain a whole music title (typically 10 minutes). The music title is considered to be broadcast between horizon v_i and horizon $v_{i+k''}$. Let us note that since the reference only contains a portion of the music title, the estimated start and end dates are not expected to match the actual (annotated) start and end dates of the title. More precisely, the estimated

dates are expected to lie within the broadcast of the title and not necessarily at the boundaries of the latter. The estimated length of the broadcast is naturally $t_f(v_{i+k''}) - t_0(v_i)$.

According to the evaluation protocol, the algorithm has to check that the reference is broadcast for more than 30s. If so, it has to output a detection stating the identifier of the reference as well as the time of detection. We define:

$$\mathcal{E} = \{v_k \ / \ i < k < i + k'' \text{and } f(v_k) = m_0\}$$

The detection time d_{out} that is output by our tracking module is:

$$d_{out} = \underset{v_k \in \mathcal{E}}{\text{median}}\{t_0(v_k)\}$$

This definition ensures that the detection date is located near the highest concentration of vote horizons having m_0 as output. This way of computing the detection date maximises the likelihood of outputting a date that actually lies within the broadcast of the title, even if there is a mistake in the estimation of the start ($t_o(v_i)$) and/or the end of the title ($t_f(v_{i+k''})$).

12.2 Robustifying the Method against Pitch-Shifting

We have seen that the method *Shazam*^{03} is not robust to pitch-shifting (see section 11.5). Our analysis has shown that this weakness comes from the fingerprint model. This is why we propose here a different fingerprint model.

Shazam^{03} uses discrete Fourier transforms in order to compute the spectrogram on which the fingerprint is based. The Fourier transform is a tool that gives the amount of energy of a signal in each frequency band. The frequency bands of the Fourier transform are linearly spaced. Yet, this spacing is not well adapted to the acoustic and musical world. Indeed, the human perception of frequencies does not follow a linear scale. This can notably be seen in the distribution of the musical notes frequencies, that are geometrically spaced. This also explains that pitch-shifting, that is perceived as a transposition (the music is heard higher or lower) physically corresponds to a multiplication of the frequencies. This mismatch between the Fourier transform and the musical perception is the reason for Brown's proposition of a Constant Q Transform (CQT) [Bro91, BP92]. Similarly to the Fourier transform, the CQT gives the amount of energy of a signal in each frequency band. However the CQT bands are geometrically spaced. It is interesting to note that in the CQT domain, pitch-shifting becomes a translation. A signal having energy in the frequency band b will have its energy moved to the frequency band $b + \kappa$ when pitch-shifted.

The fingerprint we suggest relies on a spectrogram that is obtained by the concatenation

of successive CQT transforms. Let us note that when dealing with Contant Q Transforms, we do not think in terms of 'frequencies' but rather 'frequency bins' (that are centered around geometrically -or logarithmically- spaced frequencies). We use a Constant Q Transform with 36 bins per octave (i.e. 3 bins per note) and a 10ms hop size.

The following steps are similar to the ones of the preceding method. The CQT spectrogram is tiled with rectangles of size Δt seconds and ΔB bins of frequencies (typical values are $\Delta t = 0.4s$ and $\Delta B = 18$bins). In each rectangle, the maximum point is set to 1 and all the others to 0. The result is a sparse binary CQT spectrogram. The algorithm then extracts all pairs of active points that match the pruning constraint. For two points with coordinates (t_1, b_1) (t_2, b_2) in the CQT spectrogram, the pruning constraint is given by: $t_2 - t_1 \leq \Delta t_{max}$ and $b_2 - b_1 \leq \Delta b_{max}$. The pairs of points are finally used as indexing keys. The encoding that we suggest for one pair of points is the following:

$$[\widehat{b_1}; b_2 - b_1; t_2 - t_1]$$

with $\widehat{b_1} = \left\lfloor \frac{b_1}{6} \right\rfloor$, a sub-resolved version of b_1.

As in $Shazam^{03}$, the idea is to build a representation that contains three complementary pieces of information. Though, we did choose a different set of variables. Our concern is to select pieces of information that are robust to distortions, and particularly to pitch-shifting. The first component of our representation is meant to give a rough frequency location of the pair of points. To do so, we can use the frequency information of the first point b_1 (this could equivalently be done by using b_2 or a combination, like the mean, of b_1 and b_2). However, b_1, as such, has a resolution that is too accurate to be robust to even low levels of pitch-shifting. A common way of robustifying the features is simply to sub-resolve them. This generally comes at the cost of the discrimination power of the representation. In our case, this should not be a problem since we have two more components in the representation. Finally, what we suggest as the first component is a sub-resolved version of b_1. Taking into account the resolution of our CQT, a sub-resolution by a factor 6 leads to sub-resolved bins that cover 2 successive notes of the chromatic scale. Experimentally, this has proved to be a suitable choice for common pitch-shifting ratios.

The second component is the frequency extent of the pair of points. Our two first components $[\widehat{b_1}; b_2 - b_1]$ are somehow the image of the two first components $[f_1; f_2]$ of $Shazam^{03}$. The information contained in these sets of two components can be seen as combinations of two orthogonal information variables that are: the absolute frequency of the pair, the frequency difference (that is a relative information) of the points. The main difference between our representation and the one of $Shazam^{03}$ is that we did isolate the absolute information

(in the first component) in order to sub-resolve it.

The last component is the time extent of the pair. We keep the exact same representation as in $Shazam^{03}$. As it only contains a relative time information (the absolute information is completely discarded from the representation), it is robust to cropping. Besides, we have seen that, provided we use a sufficiently low resolution for the representation of $t_2 - t_1$, this quantity is robust to time-stretching. We consequently suggest to keep the same third component, with the same resolution.

In terms of robustness, this representation has inherited the benefits of the one from $Shazam^{03}$ (robustness to additive noise, equalisation, amplitude compression, cropping). This comes from the conservation of the following principles: extraction of the points of maximum energy, binarisation of the spectrogram, use of a density criterion in frequency and time, elimination of the absolute time information. Besides, this representation has been designed in order to get an additional robustness to pitch-shifting. A signal that has a pair of points of coordinates (t_1, b_1) and (t_2, b_2) will have them moved at $(t_1, b_1 + \kappa)$ and $(t_2, b_2 + \kappa)$ in its pitch-shifted version. The encoded pair thus becomes

$$[\widehat{b_1 + \kappa}; (b_2 + \kappa) - (b_1 + \kappa); t_2 - t_1]$$

Provided the low resolution that is used to represent the first component, one can predict that, in most cases, we have:

$$\widehat{b_1 + \kappa} = \widehat{b_1}$$

It is worth mentioning that pitch-shifting will still move some values close to the border of one sub-resolved bin to the next. However, similarly to Wang's methodology, an exact matching of all pairs is not required. Indeed, the histogram step described in section 11.3 only requires that the majority of the pairs are preserved. Since the sub-resolved bins are quite large, we consider that the proportion of border values will be statistically too low to prevent a correct identification.

We can see that the second component has also been preserved:

$$(b_2 + \kappa) - (b_1 + \kappa) = b_2 - b_1$$

The encoded pair is thus kept similar in the context of pitch-shifting. Let us finally recall that the associated time-stretching effect, that would be obtained in the context of a *sample rate conversion*, has been shown to be absorbed by the resolution used for the representation of $t_2 - t_1$ (see section 11.5). This model consequently seems promising for an enhanced robustness to pitch-shifting, time-stretching or sample rate conversion.

12.3 Lowering the Complexity of the Processing

When we did introduce the notion of indexing keys in section 8.1.4, we made an analogy with a book's glossary. The words are referenced with pointers to the pages that contain these words. Similarly, in fingerprinting, the keys are referenced with pointers to the reference music titles that contain these keys. However, one should observe that a glossary does not generally contain all the words of the book. Only the most relevant and specific words are selected. By this, we mean that all articles, prepositions and adverbs are discarded. Conversely, if one very specific and technical term is used in some precise sections of the book, this term is a very good candidate for the glossary. We can thus legitimately wonder if the situation transposes to the context of audio-fingerprinting. The question that arises is then: "are there keys that are present in virtually all references?". If so, these keys would bear very little information for the identification process and would better be put aside in the search step.

One way to get an insight of the situation is to study the distribution of the number of pointed references per key. If we take an evaluation database with its set of extracted keys, we can evaluate for one key the number of pointed references. This corresponds to one realisation of the random variable *number of pointed references*. If we repeat the operation with all the extracted keys, we can store the realisations in a histogram that, once normalised, is an estimate of the probability density function of the random variable *number of pointed references*. Figure 12.1 shows such a histogram. It has been computed on a database con-

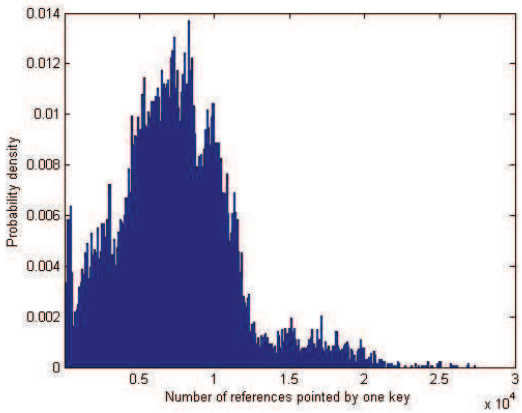


Figure 12.1: Histogram representation of the number of pointed references per key. The keys are extracted from a 30,000 titles database. The histogram bin located on abscissa x has a value proportional to the number of keys pointing to x different references.

taining 30,000 references corresponding to 60s-long excerpts of music titles. The number of

keys that have been extracted in these music titles is 13,000. We can see on the histogram that a meaningful proportion of the keys do point to a very high number of references. As said before, it seems reasonable to think that these keys do not serve efficiency in the search step. We consequently suggest an optional pruning step meant to decrease the complexity of the overall processing.

Works in the domain of text information retrieval have proposed different numerical statistics in order to evaluate the relevance of one word in a whole corpus [BYRN99]. We can notably think of the *tf-idf* statistic, whose simplest form is given for one term t and one document d of a collection $\mathcal{D}$ of documents by the product of the term frequency in the document (i.e. its number of occurrences) and the inverse document frequency ($\log \frac{\text{card}(\mathcal{D})}{\text{card}(\{d \in D / t \in d\})}$). In our work we have tried different combinations and variations of these statistics in order to elaborate a suitable pruning strategy. Our experiments have finally led us to the following definition of significance. It is quite simple and natural. Though, it has proved to work better than more complex formulae.

For each key k extracted in the references of the database, we name N_k the number of references in which the key appears at least once. N being the total number of references, we define the significance of a key k by:

$$s(k) = \frac{N - N_k}{N} \tag{12.1}$$

Basically, a key which appears in many references has a low significance. Conversely, a key which is rare has a high significance.

Let us recall that, in the proposed method, when a key is extracted from an unknown excerpt, all references possessing this key have their histograms updated. Since a key with a low significance appears in many references, it induces a meaningful number of updates and so a meaningful computation cost. Statistically, this key is also more likely to be found in any unknown excerpt (since it is a very common key). The conclusion is that a key with a low significance, due to these two leverages, induces significantly more computations than a key with a high significance.

Pruning the database consists of, for a given threshold T_{prune}, erasing from the database all the keys verifying $s(k) < T_{prune}$. When doing so, we suppose that for any reference there will be a sufficient number of keys preserved in order to ensure a correct identification. This, of course, depends on the statistical distribution of the keys and on the selected threshold T_{prune}. We have experimentally verified that the use of a reasonable threshold leads to a significant complexity gain while keeping similar performances (see section 12.4.4).

12.4 Experiments and Results

To evaluate the improvements over the original method, we keep the testing conditions similar to the ones presented in 11.4. The system is still preceded by a framing module which slices the continuous broadcast in analysis frames of length L_a and overlap o_a.

12.4.1 Proof of Concept

This first experiment we suggest is based on the PoC experiment presented in section 11.4.2. The overlap rate is set to 0 (no overlap between the successive analysis frames). The input broadcast consists of the concatenation of the references from the database. The concatenation is built in order to ensure that each analysis frame contains one and only one reference.

The modifications we brought to the initial fingerprint model are meant to bring robustness in the context of sample rate conversion, hence the idea of a modified version of the PoC. We still build a broadcast as a concatenation of the references but in this modified version, the references are firstly distorted (with a sample rate conversion) before the concatenation step. This experiment has the advantage of isolating the problem of the robustness of the fingerprint model to this particular distortion. The scores are indeed given on a frame-by-frame basis (no influence of the post-processing) and it is possible to control the level of distortion.

PoC	True Positives	False Alarms
No sample rate conversion		
Shazam^{03} [Wan03]	99.9%	0.1%
Improved (CQT-based)	99.9%	0.1%
Sample rate conversion of 1%		
Shazam^{03} [Wan03]	61.7%	38.3%
Improved (CQT-based)	95.8%	4.2%
Sample rate conversion of 4%		
Shazam^{03} [Wan03]	1.4%	98.6%
Improved (CQT-based)	84.7%	15.3%

Table 12.1: Results in the PoC experiment. The algorithms have to identify frames that are extracted from the reference database. Three different configurations are tested with different ratios (0%, 1%, 4%) of sample rate conversion. This distortion is applied to the frames that are to be identified.

The results presented in Table 12.1 seem to show that we reached our goal. The new fingerprint is still discriminative in the initial PoC experiment. Besides, we can see that it stays reliable in the presence of sample rate conversion, as opposed to the initial model whose results collapse when the distortion ratio increases.

12.4.2 Real-world comparative evaluation

This leads us to the conduction of real-world experiments. As a comparative experiment, we use the same protocols and corpuses as the real-world experiment presented in 11.4. The database contains 7,300 references corresponding to 60s-long music excerpts and the broadcast consists of 7 days of the French radio 'RTL'. Since the experiment corpus is the same as in the 2010 Quaero evaluation, we can still display the performance of IRCAM's algorithm [RP11] on this task.

$Quaero_{detec}$	Detected titles / Total	False Alarms
IRCAM [RP11]	445 / 459 (=96.9%)	2
$Shazam^{03}$ [Wan03]	381 / 459 (=83.0%)	0
Improved (CQT-based)	447 / 459 (=97.4%)	0

Table 12.2: Results with our improved method in the Quaero evaluation. The first column shows the number of broadcasts of reference titles detected by the algorithm over the total number to detect. The second column shows the number of wrongly output detections.

We can see in Table 12.2 that the detection ratio is much higher with the modified fingerprint than the original one. As far as we can tell, this confirms the fact that the missed detection from the original algorithm mainly occurred in the context of pitch-shifting. These results finally show that, in addition to being robust to the same distortions as the original model, the modified fingerprint has an increased robustness to pitch-shifting.

12.4.3 Real-world evaluation of scalability

Finally, we have led a wide-scale experiment in order to validate the scalability of the method. The framework is the same as in the previous experiment, but we now run the algorithm with a much larger database of references. In this experiment the stream is made of 5 days of radio broadcast coming from 2 different French radio stations ('RTL', 'Virgin Radio'). The references set is much larger as it contains 30,000 songs. Besides, the second Quaero protocol is used in this experiment. This means that the algorithm has to discard the broadcast of references that are shorter than 30s. This experiments thus brings into play the tracking module described in section 12.1.

The results clearly show that the algorithm is scalable. It has achieved a detection performance which is comparable to its performance in the first experiment. Though, the references database is more than 4 times larger in this experiment. It is particularly noticeable that in spite of the enlargement of the database, the system has still not output any false alarm. The multiplication of the songs in the database had yet highly increased the risk of

$Quaero_{filter}$	Detected titles / Total	False Alarms
Improved (CQT-based)	496 / 506 (=98.0%)	0

Table 12.3: Results with our improved method on the $Quaero_{filter}$ evaluation. In this experiment, the reference database contains 30000 titles. The stability of the results in spite of the increase of the database shows the scalability of the method.

having close fingerprints for different songs. As far as the detection performance is concerned, the results of this experiment show that the algorithm we propose has the ability to handle industrial sized databases.

12.4.4 Runtime

We will give here some figures about the processing times of the algorithms. These figures are given on the basis of our Matlab R 64-bits implementations, running on an Intel Core 2 Duo @ 3,16 GHz with 6MB of Cache and 8GB of RAM. We are aware that these figures give no absolute truth, since the processing times highly depend on the machines, the programming language and the optimisation of the code. They nevertheless give an order of magnitude of the runtimes with such a configuration. Besides, they allow a comparison of the different algorithms since all running times are given on the same basis. Our implementation of $Shazam^{03}$ has a processing time of 0.08s per second of signal. The improved (CQT-based) version has a processing time of 0.43 seconds per second of signal. The difference mainly comes from the extra time required for the calculation of the constant Q transform. If we apply the pruning technique described in section 12.3 with $T_{prune} = 0.5$, we obtain a speed-up factor of 35%. This reduces the processing time of the second algorithm to 0.28 seconds per second of signal with the exact same identification score. This shows that the suggested pruning technique with a reasonable threshold leads to a significant complexity gain while keeping similar performances.

We should also mention that we have led another set of experiments regarding the pruning technique. These experiments were led at a small scale and aimed at determining the influence of the pruning on the discrimination power of the method. Our preliminary tests showed that pruning the database could in fact improve the discrimination power. Thanks to the pruning technique, we could indeed prevent some false alarms in some specific cases (notably on speech sections) and also increase the true positives rate. This positive effect has not been observed in the large scale experiments since the post-processing and tracking aspects are hiding the variations that occur at a frame level. However, our belief is that such pruning techniques can improve the results in some particular conditions and use-cases.

12.4.5 Tracking

Concerning the tracking module, it filled its role perfectly since the algorithm did not output any broadcast that was shorter than 30s. Besides, no missed detection was due to a failure of the tracking module.

However, our finer experiments have put the light on a loophole in our tracking method. This loophole is linked to a specific practice that is mostly used in "young" radio channels. The latter sometimes broadcast a meaningful bit of a music title (between 15 and 30s) in order to announce the actual broadcast of the title a bit later. The typical scheme is: excerpt of the title (15 to 30s), news (1 or 2 minutes), actual title. It is clear that with this scenario, the output of the algorithm will be composed of a succession of empty sets and the identifier of the title. The problem is that this output structure also corresponds to the output structure that is expected during the sole broadcast of the music title (see section 12.1). In other words, provided the information that is available at this level of representation, the two situations (announcement + news + music title versus music title on its own) are indistinguishable. As a result, the tracking method mistakenly takes the start of the announcement as the start of the music title. The length of the broadcast of the title is consequently overestimated. However, since the detection date is computed as a median (see section 12.1), the latter will be correctly located, i.e. within the actual music title. In the end, as the title is detected and the detection date is correctly located, this loophole has no impact on the results obtained in the Quaero evaluation.

Chapter 13

Another Application of Indexed Fingerprinting

13.1 What is Recurrent Motives Detection?

In the domain of the study of multimedia broadcasts, Herley [Her06] has brought to the attention of the community the specific problem of the detection of recurrent motives. The motivation comes from the observation of the broadcasts of commercial multimedia channels, such as TV or radio channels. These streams usually contain a high level of redundancy. For example the ads broadcast by these channels are repeated a numerous number of times per day. The news channels usually broadcast the same news report every hour. As far as music channels are concerned, when a new song is released they tend to broadcast it several times a day during several months. More occasionally, an old program or a specific interview can be re-aired on special occasions. On a smaller scale, one can note the presence of systematic jingles before some specific sections of broadcasts (advertising, news, a specific program...). As we can see, the repeated objects vary in nature (news, jingle, music...), length (a couple of seconds for a jingle, more than one hour for an entire program), spacing between the different occurrences (every hour for an ad, every half a day for a new song, once in a while for an old program). In the end, the innovation rate of the stream is quite low and the latter essentially consists of a concatenation of some new items and a meaningful proportion of items that have already been broadcast.

Automatically detecting these repeating sections of broadcasts can serve several applications. The first, low-level, application we can think of is compression. Provided the streams are quite repetitive, it indeed seems a waste of space to store the entire stream. Using the detection of repeated sections can lead to a tremendous gain of space. Let us note that multimedia streams often need to be stored for a meaningful time duration because of legal

requirements. Among the applications, we can also think of automatic segmentation. Detecting the repeating structures indeed gives a sketch of the segmentation of the stream. Besides, if an automatic system is able to locate the jingles, it will be able to infer the locations of the advertisements sections or the news sections. We can thus see that automatic detection of recurrent motives can lead to automatic classification. An application with a narrower functional perimeter consists of coupling a system for automatically detecting the recurrent motives with a traditional automatic annotation system. The interest of the coupling is then to locate repeating bits of streams that do not belong to the database of the annotation system. If the repeating section fits some prerequisites (for example, if it has the characteristic length of a music title), the system can warn the operator that there is a new repeating item that is likely to be a music title and that does not belong to the current database. If it indeed corresponds to a song release, the operator can then update the database.

The problem is traditionally handled thanks to similarity calculations such as correlation [Her06] or Dynamic Time Warping [MGB11]. A similarity score is calculated for each new bit of stream with the entire past stream. If the score exceeds a certain threshold, the new bit of stream is considered to be a repeated object. The similarity calculation can be applied directly on the signal or on spectral features extracted from the signal. The problem that these methods face is the cost of calculating the similarity of a new bit of stream with the entire past stream. Let us indeed recall that the spacing between the repeated objects may vary a lot. As a result, a minimum 'memory' that is required for such systems is usually one day (meaning that if the new bit of stream is a copy of a bit of stream that has been broadcast less than 24 hours before, the system should detect it). Correlating or dynamically aligning one bit of signal with 24 hours of reference signal obviously has a high computation cost. In order to enlarge the scalability of their systems, the authors propose the use of downsampling methods or bufferisation techniques.

Still, one efficient and nice way to deal with this scalability issue would be the introduction of an indexing technique. Between this paradigm and the idea of diverting a fingerprint method from its traditional goal of automatic annotation of a broadcast, there is a fine line. Provided the setup of some adaptations, we will see that a fingerprint-based approach can perfectly handle the use-case of automatic detection of recurrent motives. Let us note that this idea of fingerprint-based systems for the recurrent motives use-case has been exploited in the works of Burges [BDP+05] and Ogle [OE07]. However, very few technical details are given in [BDP+05], whereas in [OE07] the problem is treated in an offline setting. Conversely, we give in the following sections a precise description of the architecture that allows to go from an automatic annotation system to an online system for the detection of recurrent motives in a stream.

13.2 A Specific Task in Quaero

A specific evaluation has been created for the detection of recurrent motives in a stream. In order to exploit the corpuses that we have at our disposal in the context of the automatic annotation use-case (see section 9.3.2), the focus has been set on the particular use-case "detection of repeating music titles in a broadcast". However, since the aim of the evaluation is solely to assess the ability of a technology to detect the redundancy and not the ability to classify (musical versus non musical signals), the task of the algorithm has been defined as "in a given stream, find any pattern which lasts at least 90 seconds and which can be found at least twice in the stream". In the corpuses that we have available, the set of recurrent motives that are longer than 90s indeed matches the set of broadcast music titles.

The specificity of our work is that we did include a cross-radio corpus in the evaluation. In the state of the art, authors usually focus on an evaluation framework based on a stream coming from one single channel. This means that the repeated items are very likely to be broadcast with the exact same post-processing (same equalisation, compression, pitch-shifting...). In the Quaero evaluation, two corpuses have been created. The *Test3Days* corpus is composed of 3 consecutive days of the radio 'Nostalgie'. The *Test3Radios* corpus is composed of 3 simultaneous days from the 3 radios: 'NRJ', 'Virgin Radio' and 'Chérie FM'. Since some motives are repeated from one channel to the other, it is very likely that this second corpus contains recurrent motives that have been broadcast with different post-processings. As we can see, the memory capacity that is expected from the system is also quite challenging as it corresponds to 3 days of broadcast in both corpuses.

13.3 How to adapt a Fingerprint System to this Use-Case

The general idea of the diversion of an automatic annotation system is the following. What a traditional fingerprint system does when it analyses an unknown excerpt is looking for the best match to this excerpt in a database. What we have to do in order to turn this system into a recurrent motives detection system is replacing the content of the reference database by the past stream. So, the system will look for the best match to the analysed excerpt in the past stream. We thus see that such a system has two simultaneous tasks to accomplish when analysing a stream segment: looking for the best match for this segment in the past stream, storing this new segment of stream in the database since it will be part of the past stream when analysing the next segment of stream. However, we have to stay aware that the analysed segment is possibly a repetition of a past section of the stream. In this case, it

is not desirable to store the current stream segment, which is a duplicate, in the database. The storage mechanism therefore has to take into account the possible detection in the past stream of a match before writing the analysed segment in the database.

We give at first a general description of the corresponding architecture, shown in Figure 13.1. The different building blocks of this architecture are described below.

13.3.1 General architecture

The stream is framed and then linearly processed. Each frame undergoes a fingerprint extraction. From here, the system forks. One branch is dedicated to analysing the fingerprint (in practical terms, looking for matches in the past stream), the other is dedicated to storing the fingerprint in the database containing the past fingerprints.

In the analysis branch, the fingerprint is matched against the database containing the previous frames' fingerprints. The identification scheme that we have proposed in the previous sections relies on a post-processing step that fuses the matching results of successive analysis frames. This process has indeed shown to be a reliable way to eliminate the false alarms while keeping a high detection ratio. For this reason, the matching result of the current frame is combined with the matching results of the P previous frames. Based on these matching results, a repetition detection decision is taken. In case of a detected repetition, the storage branch is updated so that it will not store repeated frames in the database.

As we can see, it cannot be decided whether the current analysis frame is a repetition of a previous section of stream before processing P further frames. This justifies the fact that in the storage branch, the frame's fingerprint is pushed into a FIFO (First In First Out) buffer. This buffer delays subsequent storage processing for this frame. As the current frame fingerprint enters the buffer, the last fingerprint of the FIFO buffer is pushed into the database. Though, this latter will be written only if it has not been detected as part of a repeating segment.

13.3.2 Framing and fingerprinting

As in the preceding algorithms, the input stream is sliced in *analysis frames* f_n ($n \in \mathbf{N}$) of length L_a. However, as we are dealing with a system that is meant to detect redundancy, we avoid introducing unnecessary redundancy whenever possible. For this reason, we suggest a non-overlapping framing (in other words, $o_a = 0$). A typical value for L_a is 5s.

Each analysis frame then has its fingerprint extracted according to the methodology described in section 12.2. In short, the CQT-based spectrogram of the signal is computed. Local maxima of this spectrogram are set to 1 whereas all other points are set to 0, thus

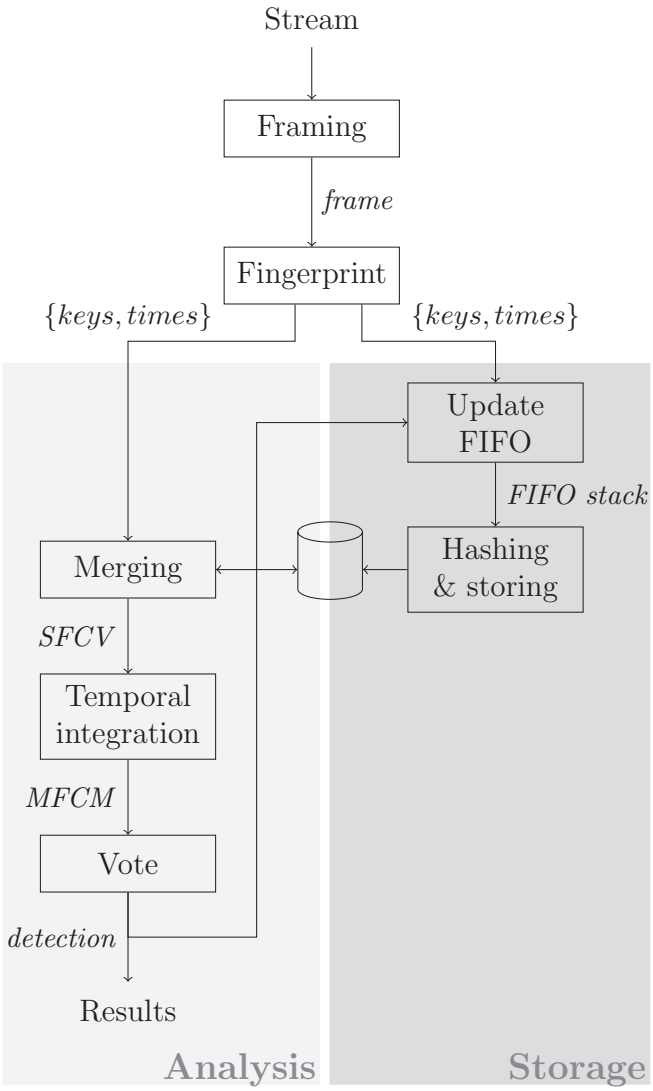


Figure 13.1: Architecture of the fingerprint-based system for recurrent motives detection in an online stream. When analysing an incoming bit of stream, the system has to simultaneously look for a match in the past stream and manage its storage in the database.

leading to a binary CQT-spectrogram. Points set to 1 in the binary spectrogram are then grouped in pairs. Each pair made of two points with coordinates (t_1, b_1) (t_2, b_2) in the spectrogram is encoded with the following vector:

$$[\widehat{b_1}; b_2 - b_1; t_2 - t_1]$$

This vector is finally used as an index key. For a given analysis frame, the fingerprint module thus outputs a set of keys along with their dates of occurrence. Formally, we define $\mathcal{K}$ the set of keys extracted in f_n. Let $O_k(f_n)$ be the number of occurrences of the key k in f_n. We then define $t_k(f_n) = \{t_k^i(f_n)\}_{i=1..O_k(f_n)}$ the set of times of occurrence of the key k in the frame f_n. The output of the fingerprint module is $\{(k, t_k(f_n))\}_{\forall k \in \mathcal{K}}$.

Let us note here that there is a slight difference of notation with the preceding system. The times of occurrence $t_k^i(f_n)$ are now given with respect to a global reference (time of occurrence of the key *in the stream*) whereas they were given with respect to a relative reference before (time of occurrence of the key *in the analysis frame*). The two formulations are theoretically equivalent: it suffices to translate the date with the time of occurrence of the analysis frame in the stream. However, we will see that using absolute time quantitites makes more sense in the present use-case, which is exclusively stream-based (even the reference database is a stream). It notably reads much simpler when formulating the post-processing step.

The database contains all the keys that have been extracted in the past stream with their times of occurrence in the stream. As it is meant to represent the past of the stream, we use the notation f_{-1} to refer to the database. Consequently, a key k appears $O_k(f_{-1})$ times in the database at times of occurrence $\{t_k^i(f_{-1})\}_{i=1..O_k(f_{-1})}$. When querying the database with key k, we get in output $\{t_k^i(f_{-1})\}_{i=1..O_k(f_{-1})}$. As in the other implementations, we use the database engine Berkeley DB set to its Hash Table mode.

13.3.3 Merging the database outputs

The analysis branch starts with the merging step, which is similar to the one described in 11.2. It aims at finding the closest match to the current analysis frame in the stream. The main idea is that if the current frame is the repetition of a previous section of the stream, its keys should all be stored in the database. Furthermore, all the keys extracted from the analysis frame should be retrieved in the past with the same delay. We then adopt the following methodology to find the best candidate in the past.

We compute the set of differences

$$\left\{\{t_k^i(f_n) - t_k^j(f_{-1})\}_{\forall (i,j) \in [\![1;O_k(f_n)]\!] \times [\![1;O_k(f_{-1})]\!]}\right\}_{\forall k \in \mathcal{K}}$$

We store these time differences in a histogram. The highest peak in the histogram gives the best candidate delay for a repetition.

Let us recall that this methodology ensures the retrieval of the best candidate. Though, it does not require that all keys are preserved from one version to the other. It only requires that the majority of the keys are preserved. This makes the method robust to distortions that would corrupt part of the keys between the two versions.

13.3.4 Enhanced post-processing

We should note here that this use-case is more demanding than the initial one in terms of identification efficiency. We were indeed dealing with a detection use-case when working in the automatic annotation task (the system only had to output an isolated detection of the broadcast reference item) whereas in this use-case, the system has to entirely locate the repeated section of stream. For this reason, we took advantage of this use-case to enhance the post-processing scheme. This enhanced scheme should minimise the risks of missing detections at the analysis frame's level.

This enhanced post-processing step requires, in input, the M best candidates for each analysis frame. As a result, the output of the merging step is the M best candidate delays. The system stores the matching results of several successive analysis frames before making a detection decision. The merging step outputs a vector of M best candidates (that we call *Single Frame Candidates Vector* - SFCV) that is integrated in a $P \times M$ matrix that contains the P last SFCVs. We call this matrix the *Multiple Frames Candidates Matrix* - MFCM.

Provided the use of absolute times of occurrences, the post-processing based on the fusion of local decisions described in 11.3 can be generalised to the case of M best candidates in the following way. For a given detection threshold T_{vote}, let M_n be the MFCM after integrating the matching results of frame f_n. Let $\mathcal{C}$ be the set of candidate delays that appear in M_n and h be a function that counts the number of occurrences of a candidate in the MFCM. Let δ be a function that is defined by:

$$\delta(x, y) = \begin{cases} x & \text{if } y > T_{vote} \\ \emptyset & \text{otherwise} \end{cases}$$

The vote function is then defined by:

$$v(M_n) = \delta(\underset{c \in \mathcal{C}}{\operatorname{argmax}}\{h(c)\}, \max_{c \in \mathcal{C}}\{h(c)\})$$

However, this vote model may generate some instability when dealing with objects that contain an inherent repetitive structure. For example, let us imagine that the database

contains one song with two similar choruses. As a consequence, when processing an analysis frame belonging to a chorus of the same song, the two choruses in the database will be candidates in the MFCM. Besides, their number of occurrences in the MFCM will be very close. The result is that, when processing the successive analysis frames of a chorus, the detections issued by the post-processing module may look like that:

`chorus1-chorus2-chorus1-chorus1-chorus2...`

This, of course, is not desirable, since we would like our algorithm to consider that the successive analysis frames all belong to the same chorus. Ideally, we would like the algorithm to detect `chorus1` when processing the first analysis frames containing a chorus and `chorus2` later on in the stream.

In order to achieve this goal, we modify the preceding vote model so that it becomes autoregressive. The autoregressive aspect is obtained by favouring the delay that best corresponds to the preceding vote result. This ensures a certain continuity in the algorithm detections. So, if the start of the song has been detected, and when reaching the chorus, the algorithm will naturally tend to select the first chorus of the song in the database. Formally it consists of replacing function h in the vote by $\widetilde{h}$, which is defined by:

$$\widetilde{h}(c) = \begin{cases} h(c) + \beta & \text{if } v(M_{n-1}) = c \\ h(c) & \text{otherwise} \end{cases}$$

In our implementation, $\beta = 1$.

13.3.5 Storage

One of the principles of the architecture is to store the fingerprints of the analysis frames that have been processed in a database. However, we do not wish to store in this database the fingerprints of the frames that are detected as repetitions. There are two reasons for that. First, it would be a waste of space. Second, when matching a third fingerprint that would be alike, we would obtain two good candidates instead of one. That would uselessly jam the matching process.

As we have seen, the algorithm requires the matching results of P analysis frames before being able to make a decision. This is why we store in a temporary FIFO buffer the fingerprints of the processed analysis frames. This buffer contains $B > P$ processed fingerprints. If further processing outputs a repetition detection for frame f_n, its fingerprint in the FIFO buffer is labeled so that it will not be written in the database.

The FIFO buffer also has a screening function. Indeed, the repetitions that occur before

B frames are not detected (since the corresponding fingerprints have not been added to the database). Depending on the use case, this can be useful to prevent over-segmentation. For instance, when segmenting a radio broadcast, one would usually want repeating segments that correspond to whole songs. Though, if there is no screening and if the songs contain repetitive choruses, the algorithm might annotate the songs in several repeating bits (the choruses) and unrepeated bits (the verses). By setting B to a larger value than the length of the song, we can prevent the system from detecting repetitions within the song.

13.3.6 Output of the algorithm

The framework outputs a decision for each analysis frame. It is either considered as a repetition of a previous frame, or as a first occurrence.

This output can be represented in a plan with the following methodology. For each analysis frame, we define a dot with coordinates (t_1, t_2). t_1 is taken equal to the time of occurrence of the analysis frame in the stream. If the frame is a repetition of a past frame, t_2 is taken equal to the date of the original frame in the stream. Otherwise, we take $t_2 = t_1$.

A graphical illustration of the result is given in Figure 13.2. The origin point is the experiment starting date. At this point the database is empty so no repetition can be found. Points on the diagonal indicate frames detected as first occurrences. Points outside of the diagonal indicate repeated frames. Alternatively, the output can be considered as a binary self-similarity matrix of the streamed data. In the shown example, one can see nine repeated sections of broadcast. Their lengths suggest that they correspond to the repetitions of music titles.

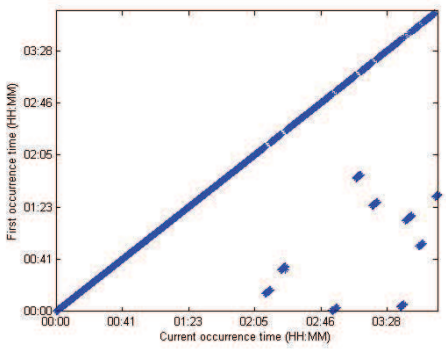


Figure 13.2: Graphical illustration of the output of the algorithm in the fashion of a similariy matrix. On this example, one can see that the signal broadcast between 02:11 and 02:14 is a repetition of the signal from 00:14 to 00:17.

13.3.7 Tracking

As stated in section 13.2, the algorithm's task is to detect repeated sections that are longer than 90s. Consequently, as in the $Quaero_{filter}$ evaluation, there is a need to setup a tracking mechanism in order to evaluate the length of the detected repetitions. The tracking methodology that we suggest is based on a clustering paradigm that apply to the output described in the previous section. As in the first use-case, we tried to design a tracking method that is robust to the possible imprecisions that can occur in the detection phase.

In detail, the clustering strategy is the following. The algorithm only considers the dots that are outside the diagonal. Let $d_1, d_2, ..., d_n$ be these dots, sorted in the order of increasing abscissa. By construction, we know that the abscissa are strictly increasing. One cluster $\mathcal{C}_i$ is fully defined by the set of dots d_k composing it. In this set we call the border dot $d_{\mathcal{C}_i}$ the one with the largest abscissa. By construction, it is also the dot with the largest index.

$$k_{\mathcal{C}_i} = \max\{k \,/\, d_k \in \mathcal{C}_i\}$$

The clustering strategy is given by the following pseudo-code.

```
// Init
i = 0;
// Main loop
while ∃ k / ∀ l  d_k ∉ C_l do
    // Start a new cluster
    i = i + 1 ;
    C_i = {d_{k_0}} with k_0 = min{k / ∀ l  d_k ∉ C_l};
    // Expansion of the cluster
    for j from k_{C_i} + 1 to n do
        if d(d_{k_{C_i}}, d_j) < Th then
            C_i = C_i ∪ {d_j} ;
        end
    end
end
```

The algorithm builds the clusters one after another. It starts with one cluster $\mathcal{C}_1$ that contains the single point d_1. This first cluster is then expanded, from left to right. To this end, all the dots $d_2, d_3, ..., d_n$ are iteratively tested. If the tested dot lies at a distance smaller than Th from the current border point of the cluster, it is added to the cluster. In our implementation we use the norm L_∞ as the distance function. As a consequence, we can stop the inner loop (**for** loop) as soon as $x_j > x_{\mathcal{C}_i} + Th$. Once the cluster has been expanded to its maximum size, we start a new cluster. This new cluster is initialised with the first dot of the list that

has not yet been included in any cluster.

In the end, we consider that each cluster corresponds to a repeated section of broadcast that goes from $\min\{x_k\}$ to $\max\{x_k\}$. The original version is located between $\min\{y_k\}$ and $\max\{y_k\}$. The duration of the repeated section is thus: $\max\{x_k\} - \min\{x_k\}$. According to the evaluation guidelines, the algorithm does output the detection of the repetition only if: $\max\{x_k\} - \min\{x_k\} > 90s$.

13.4 Experiments and Results

13.4.1 Evaluation on a synthetic broadcast

As for the automatic annotation use-case, we have designed a first evaluation protocol that relies on a synthetic broadcast. The interest of such an evaluation is that it allows to get a score that can be estimated at a frame level. Conversely, the use of real-world broadcasts with macroscopic annotations only permits the computation of detection-based scores.

This task consists of determining for each incoming analysis frame whether it is a repetition of a previous frame. If so, the exact first occurrence in the stream is retrieved. A synthetic stream is built as a concatenation of 140 audio excerpts randomly taken from a pop song database[1]. Each excerpt lasts 30 seconds, 100 of them occur twice in the stream and the 40 remaining are not repeated. The total duration of the stream is thus 2 hours. Analysis frames are 5 seconds long, the complete dataset therefore consists of 1440 frames, 600 of which are exact repetitions of previous frames.

Synthetic	Precision	Recall
CQT-based	97.8%	95.1%

Table 13.1: Results of the algorithm on a synthetic broadcast. The algorithm has to detect the repeated analysis frames. By construction of the broadcast each analysis frame is either an exact repetition of a previous frame or a totally new frame. The Recall is the ratio of True Positives whereas the Precision is linked to the number of False Alarms.

The score is given in terms of precision and recall. Table 13.1 summarizes the obtained results. The system reaches good levels of precision and recall for this frame-based evaluation. The results confirm the relevance of the proposed architecture as a repetition detection system. We can consequently move towards the real-world evaluation.

[1] http://quaero.org

13.4.2 Real-world evaluation

As mentioned, it is virtually impossible to get a frame-by-frame repetition annotation on a real broadcast. On the other hand, the use of the annotations provided within Quaero allows a detection-based evaluation. As stated in section 13.2, the task of the algorithm is to *detect all repeating segments that are longer than 90s*. The detected repetitions should then match the repeated music titles, for which we have the manual annotations.

Two corpuses were provided. The first one consists of three days of the radio 'Nostalgie'. The second corpus consists of 3 simultaneous days from the 3 radios: 'NRJ', 'Virgin Radio' and 'Chérie FM'.

Quaero	Detected rep. / Total nb	False Alarms
Test3Days	565 / 565 (=100%)	10
Test3Radios	626 / 644 (=97.2%)	10

Table 13.2: Repeating objects detection scores for a real-world radio broadcast. The *Test3Days* corpus contains 3 days of the same channel whereas *Test3Radios* contains 3 simultaneous days from 3 channels. The latter is thus more complex since it contains the post-processing of different channels.

The results are given in Table 13.2. In the first corpus, the algorithm did not miss any detection. As expected the second corpus seems a little more difficult since it includes the post-processings applied by various radio channels. The detection ratio however stays very high. As far as our analysis of the errors could go, it seems that all of them were due to the tracking step. More precisely, including several radio channels in the same corpus has led to a situation where several edits of the same song are present. The difference between the edits can be various. In this corpus, they mostly concern the structure of the song: one edit will have an additional bridge and/or an additional introduction and/or a further verse. Sometimes, we also have additional featuring by another singer. From the point of view of the annotations, two different edits are considered the same music title. At the analysis frame level, the situation is illustrated in Figure 13.3. We see that some bits of the title are an exact copy of the previous edit whereas some other bits are new because they were not present in the initial edit. Conversely, it might happen that there is a 'jump' in the detection if the new edit skips parts of the initial edit. To some extend, our tracking methodology is able to correctly cluster one new edit in spite of these discontinuities. This is done thanks to the threshold Th. However, the few missed detections are due to edits which include too many differences with the initial one. Conversely, the false alarms are linked to the presence of small repeated portions of stream that have been mistakenly clustered together. There is of course a trade off in the tuning of Th.

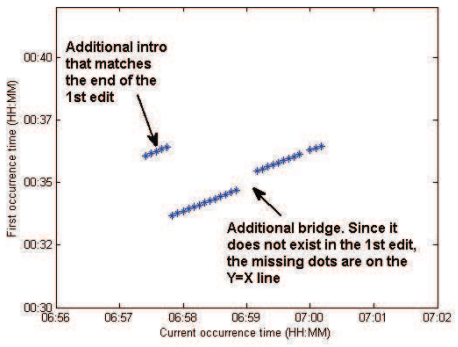


Figure 13.3: Zoom in the binary similarity matrix on the detection of a 2^{nd} edit while the database contains the 1^{st} edit. The 2^{nd} edit is broadcast from 06:57 to 07:00 and the initial edit from 00:33 to 00:36.

Let us finally mention that this evaluation protocol is somewhat tricky when the algorithm stores a misleading reference in the database. Let us for example figure that the first broadcast of one given music title is a very specific and distorted edit. This edit will be stored in the database. If the level of distortion goes beyond the capacities of the tracking step, the algorithm will consequently fail identifying the broadcast of the other edits. The trick is that this will then generate one additional missed detection for each broadcast of this music title. The undesirable initial storing of one specific edit is thus penalised several times.

Chapter 14

Generalisation of the Search Strategy

14.1 General Formulation

A further look at the search strategy used above reveals that it is quite uncorrelated with the fingerprint model. In fact, we will see later that this search strategy can be seen as an application of the *Basic Local Alignment Search Tool* algorithm [AGM+90]. More precisely, the only requirement for the search strategy to be applicable is that the fingerprint consists of time-localised keys.

Formally, what we need is that for a given signal u to analyse, the fingerprint module outputs a set of features along with their dates of occurrence. We call these extracted features keys. We define $\mathcal{K}$ the set of keys extracted in the signal u. Let $O_k(u)$ be the number of occurrences of the key k in u. We then define $t_k(u) = \{t_k^i(u)\}_{i=1..O_k(u)}$ the set of times of occurrence of the key k in the signal u. The output of the fingerprint module is $\{(k, t_k(u)\}_{\forall k \in \mathcal{K}}$.

Provided a fingerprint can be put under this form, the system can then compute the time differences of the times of occurrence of the keys in the signal and in the database (histogram step). The further processings such as the fusion of local decisions and the tracking step can subsequently be applied.

In order to demonstrate the general applicability of the search strategy, we have integrated our code with a different fingerprint model. The model has been deliberately chosen very different from the initial one. As far as the use-case is concerned, we have employed this second fingerprint in the context of the recurrent motives detection. This, of course, does not restrict the scope of the generalisation, which could similarly apply in the first use-case.

14.2 A Sparse-Decomposition Based Fingerprint

The fingerprint presented above is based on a peak picking mechanism in the time frequency domain. Alternatively one can build a fingerprint based on a sparse decomposition of the signal in a redundant dictionary. Let u be a framed signal $\in \mathbb{C}^N$ and Φ be a dictionary of elementary waveforms $\phi_k \in \mathbb{C}^N$ called *atoms*. We denote $\hat{u}_m$ an m-term approximant of u in Φ, that is to say a linear combination of m waveforms: $\hat{u}^m = \sum_{i=0}^{m-1} \alpha_i \phi_i$. There are many different ways of building such approximant. A fast one is to iteratively select the ϕ_i according to an energetic criterion:

$$\phi_i = \underset{\phi \in \Phi}{\operatorname{argmax}} \, |\langle u - \hat{u}^{i-1}, \phi \rangle|$$

Algorithms based on this greedy paradigm are called Matching Pursuits (MP) following the work of Mallat et al. [MZ93].

Sparse decompositions have initially been proposed for compression purposes. Indeed, in a variety of multimedia contexts, wavelet dictionaries (e.g. for images) and Fourier-based transforms (e.g. MDCT for audio) have enabled a fair amount of dimensionality reduction. The idea of exploiting sparse decompositions for fingerprinting has already been proposed (e.g. in [CB07]).

An m-term approximant $\hat{x}^u$ can efficiently be used as a fingerprint if: 1) its dimension is much lower than that of u 2) two different signals would yield significantly different fingerprints and 3) the fingerprints exhibit some robustness to mild distortions. Most MP-like algorithms are only tailored for the first of these properties. However, in a fingerprint context, we are not interested in minimizing a reconstruction error, but in maximizing a discriminating power. Therefore, two options can be considered: either build a fingerprint from an existing m-term approximant or modify the decomposition algorithm so as to only select elements that will favor good fingerprint properties in $\hat{u}^m$.

In this work we have implemented the second approach, and the following fingerprint construction is performed. We have used a multiscale MDCT dictionary and a plain MP algorithm with the additional property that atom selection in the time frequency neighborhood of previously selected atoms is discouraged. The selection criterion at iteration i becomes:

$$\phi_i = \underset{\phi \in \Phi}{\operatorname{argmax}} \, \lambda(\phi, \Phi_I).|\langle u - \sum_{j=0}^{i-1} \alpha_j \phi_j, \phi \rangle|$$

where Φ_I is the set of previously selected atoms and $\lambda(\phi, \Phi_I)$ is a binary penalty term set to zero if any previously selected atom is in the time-frequency neighborhood of ϕ.

For a given signal u, an approximant $\hat{u}^m$ is computed and the set of keys used by the fingerprinting system is simply the set of indexes of the m atoms chosen in the dictionary Φ. By limiting the decomposition to a small number m of iterations, the dimensionality can be greatly reduced. However, the fingerprint discriminative power increases with the number of atoms selected in the decomposition.

14.3 Experiments and Results

This second fingerprint has been tested with the protocol presented in section 13.2. The experiments are the same as the ones in section 13.4.

Synthetic	Precision	Recall	CPU Time - Fingerprint	CPU Time - Total	Memory
CQT-based	95.1%	97.8%	0.12s	0.20s	9.3MB
MP-based	94.5%	91.5%	0.33s	0.40s	2.4MB

Table 14.1: Results of the algorithms on a synthetic broadcast. The algorithms have to detect the repeated analysis frames. By construction of the broadcast each analysis frame is either an exact repetition of a previous frame or a totally new frame. The Recall is the ratio of True Positives whereas the Precision is linked to the number of False Alarms.

The first experiment relies on a synthetic broadcast and consists of determining for each analysis frame if it is the copy of a previous frame. The MP-based fingerprint has been computed with a dictionary of 3 MDCT scales and stopped after 150 iterations (labeled MP-150). Table 14.1 summarises the obtained results and recalls the ones obtained with the initial fingerprint. The system reaches good levels of precision and recall for this frame-based evaluation with both fingerprints. The recall with the CQT-based fingerprints is better than when using the MP atom indexes. However, it is also more memory consuming. The MP-based fingerprints are smaller, but less robust as the recall shows. The CQT-based fingerprints are faster to compute but the matching process roughly requires the same amount of time for both methods.

The results for the real-world experiment are given in Table 14.2. We see that the new fingerprint slightly underperforms the initial one in the first corpus in terms of recall. The gap however widens in the second corpus. This tends to show that this second fingerprint is less robust to the post-processing distortions. Let us indeed recall that this second corpus is indeed much more likely to include distortions between the different copies of one music title since it includes several radio channels. The number of false alarms is of the same order, and

Quaero	Detected rep. / Total nb	False Alarms
Test3Days Corpus		
CQT-based	565 / 565 (=100%)	10
MP-based	537 / 565 (=95.0%)	7
Test3Days Corpus		
CQT-based	626 / 644 (=97.2%)	10
MP-based	478 / 644 (=74.2%)	8

Table 14.2: Repeating objects detection scores for a real-world radio broadcast. The *Test3Days* corpus contains 3 days of the same channel whereas *Test3Radios* contains 3 simultaneous days from 3 channels. The latter is thus more complex since it contains the post-processing of different channels.

as said before, mostly related with the tracking strategy.

In conclusion, this work has shown the general applicability of the method. This second fingerprint has been employed on the real-world use-cases with industrial data. Even though the second model slightly underperforms, the results still seem satisfying enough to open the door to real-world applications. Let us note that the second fingerprint model could be further enhanced and robustified. One could for example consider the grouping of the atoms in pairs and use a pitch-shifting robust encoding rather than using single atoms. Interestingly, this experiment also shows that the system can be used as a test-bed for drawing comparisons between fingerprints models in the context of a real-world use-case.

Chapter 15

Conclusion

We have started this part with the description of our implementation of a method of the state of the art [Wan03]. The testing of this implementation with real data led us to the improvement of some specific aspects. The first substantial modification that we brought is the addition of an elaborated post-processing step based on the fusion of local decisions. More fundamentally, we proposed a change in the fingerprint model, which resulted in an increase robustness of the method to pitch-shifting. A second step in our work has consisted of adapting our audio-fingerprint methodology to a different use-case. We have indeed showed that index-based audio-fingerprint systems could be transformed in order to handle the detection of recurrent motives in a stream. Finally, putting things in perspective in the proposed model, we realised that the search method is quite independent of the fingerprint model. To demonstrate this, we did integrate a different kind of fingerprint in the same index scheme.

All these pieces of work were extensively tested thanks to the data, evaluation frameworks and protocols provided within Quaero. Let us recall that these evaluations were designed to faithfully reflect real-world use-cases, notably thanks to the use of industrial corpora and databases. The striking fact when looking at the results in our different experiments is that all algorithms are very efficient. The post-processing unit that we proposed adequately limits the number of false alarms and the modified fingerprint is very good at matching two equivalent signals, even in the presence of distortions. In the detection of recurrent motives use-case, our analysis showed that the few errors that were output were not due to the fingerprint algorithm. They were coming from the clustering step.

More generally, we should note that IRCAM, which took part in the Quaero work-package "audio-identification and fingerprinting", did achieve the same kind of performance in the corresponding evaluation. It is, in consequence, very tempting to infer that the technologies of the succeeding companies of the domain, such as Shazam, reach this kind of scores. In

the end, it seems that the traditional audio-identification use-case has somewhat run dry. For a given method, it will of course always be possible to find a set of sufficiently severe distortions to cause its failure. But the tuning of the parameters, the slight adaptation of the model will probably allow to overcome the difficulty.

For that matter, my point of view is that the task needs to be extended with new challenges. For example, our experimental observations showed that the model we use is suitable only for *exact matching*. As soon as we are dealing with two different recordings of one same title, the method cannot match them. This is not surprising since the features extracted in the representation are low-level features. Consequently, it cannot be expected that these will be preserved when dealing with a different signal, even if the music sounds similar. We thus decided to get into the development of methods that can handle the approximate matching use-case (the algorithm has to detect the broadcast of items that are equivalent or only similar to one reference). This constitutes the object of the next part.

Part V

Approximate Matching

Chapter 16

Introduction to Approximate Matching

In this part, we tackle the problem of approximate matching. By this, we mean that the task of the audio-fingerprint algorithm is redefined. It now has to identify any broadcast of a reference item (possibly distorted in the transmission chain) but also the broadcasts of items that are only similar to one of the references.

Within our documentary research in the field of "audio-fingerprinting", we found only one work mentioning the objective of approximate matching. The document is a patent from Google [LWR12]. It describes a fingerprint representation called "intervalgram" that is a tonality-robust summary of the chromagram around pre-determined reference times. The summary is obtained by a non-uniform temporal sampling of the chromagram and is meant to represent the melodic lines around the reference times. The search strategy is not included in the description.

Given our approximate matching use-case, it is however of interest to study the field of "cover song recognition". The latter initially consists of methods that can evaluate the musical similarity between two signals. Although one can find methods relying on the extraction of melody [Mar06] or rhythm [HS08], most of the contributions rely on the matching of chroma-based features [EP07, SGHS08, MKC05]. It must however be understood that the above-mentioned methods do not include any indexing aspect. Finding the best match to a query signal within a set of references thus comes to comparing the query to all the references. Given the proposed matching strategies (Dynamic Time Warping or in the simplest cases a linear product), there is no hope that these methods can scale up to large databases.

The domain of cover song has, however, slowly moved towards the problem of large reference databases. The work in [NKM02] marks a first step in this direction. In this work the authors propose a search speed-up for a given similarity matching. The signal

model consists of quantised chroma vectors and the search is inspired by approximate string matching. However the chosen search method is online, thus depriving the method from the efficient use of an index.

More recently, the authors of [MO08] have proposed a two-steps process for the identification of classical works independently of the performance. The first step is an index search. The index keys consist of quantised versions of the chroma vectors. The aggregation of the index outputs however stays quite simplistic (it is a simple counting of the index keys in common between the query and the references). The second step is a fine comparison of the candidates output in the first step with the query. Strangely, the fine comparison step has nothing in common with the index search (different signal model based on spectrogram peaks, different matching strategy based on Hidden Markov Models).

In [KM08], the authors propose an indexing scheme that builds upon the specific chroma-based features presented in [MKC05]. To this aim, the authors introduce a further quantisation of their initial feature on a finite codebook. During the learning stage, the quantised version of each reference is computed. Each codeword is then indexed in association with pointers to all references containing this codeword. The identification stage finally consists of computing the quantised version of the query and use the index to retrieve the references having a high number of correlated codewords with the query. As we will see later, this can be seen as an application of the BLAST (Basic Local Alignment Search Tool) strategy with a subsequence length of 1.

In 2012, the authors of [MBHF12] have proposed an approach that relies on the same principles. In their method, the features of interest are the chromas. The quantiser that is suggested is a chord-estimation technique. The search strategy explicitly relies on BLAST.

As far as chroma indexing is concerned, we can note the work in [BME11]. This paper is clearly focused on the Million Song Database [1] and its associated features, which are obtained through the Echo Nest API [2]. From the available features, the authors propose the computation of a beat-synchronous chroma representation. Starting from this representation, they suggest the extraction of indexing keys. These are computed in a way that is very similar to [Wan03]: salient peaks are extracted and grouped together to produce index keys. The identification finally consists of finding the reference that have the highest number of index keys in common with the query. The authors, however, mention that the results obtained on the Million Song Database are rather disappointing. This is how they justify the proposition of a new matching scheme in [BME12]. In this work, the authors abandon the notion of index. They propose to summarise the signal in a very compact vector that is meant to

[1] http://labrosa.ee.columbia.edu/millionsong/
[2] http://developer.echonest.com/

be discriminative enough to represent the music while being invariant between similar music pieces. These vectors correspond to the magnitude of the 2D-Fourier Transform of the beat-synchronous chromagram, further compressed by PCA. It is shown that this representation has nice invariance properties. Finally, the identification step consists of calulating the compact representation of the query and then searching for the nearest neighbours in the reference database. Since the vectors are very compact, this can be done in an exhaustive way even with a large number of references.

In the light of these works it seems clear that there is an ongoing fusion between the initially separated fields of "audio-fingerprinting" and "cover song recognition". In the following, we describe two fingerprinting methods that we developed for the approximate matching use-case. The first method that we present relies on a chords transcription scheme associated with the BLAST search strategy. The second method that we propose for approximate matching uses a mixture of rhythmic and harmonic information. As far as the first method is concerned, it should be noted that the already cited work of [MBHF12] has been published in 2012. It was thus not available at the start of our work on the chords-based strategy. We discovered *a posteriori* that this work and ours present strong similarities in their philosophies. However, there are still a number of differences between them. These differences are listed below but may be better understood after reading the detailed description of our method in the following chapter. As far as BLAST is concerned, their implementation does not feature any neighbourhood generation. Besides, the aggregation strategy (called filtering in their work) is quite different from ours. The chords estimation they use is not detailed and their method relies on a a very light dictionary of chords (each chord is represented by its sole root note). They have specifically tackled the issue of transposition (by normalising the chords sequences) whereas we have not studied this particular issue. Their study mostly focuses on the optimal selection and the optimal length of the subsequences in BLAST whereas we have put a strong emphasis on the influence of the chords model.

Chapter 17

Chords-Modelling Approach

17.1 Overview

In this chapter, we describe an audio-fingerprint method that is based on the transcription of the signal in a sequence of chords. More precisely, the chords sequence representing a given signal constitutes its fingerprint. When searching for an unknown signal, the process consists of transcribing it in a sequence of chords and then looking for the closest transcription in the reference database. However, we do not expect the transcription in chords to be a robust feature. For this reason, we set up a specific approximate search method.

The chapter is organised as follows. In the introduction, we give a contextual presentation of the chords transcription problem. This includes a precise definition of what a chord is and a quick listing of common techniques used for chords estimation. Then we move to the specific description of our method, starting with the chords transcription model that we use. This includes the computation of the low-level representation of the signal (chromagram) and the estimation of the best matching chords sequence. In the next section we underline the fact that, given our audio-identification use-case, there is a need to define a distance between the chords sequences. This distance should ideally reflect the musical similarity between the transcribed signals. The distance model that we propose is based on dynamic programming. The following section details our search strategy. The idea of this search step is to bring, in a very fast way, a set of candidates from the reference database that will be the closest to the unknown signal, in the sense of the above defined distance. The chapter ends with the presentation of a real-world evaluation framework on which our method has been extensively tested.

17.2 Introduction to Chords-Sequence Based Audio-Identification

17.2.1 Positioning of the work

The idea here is to model any musical signal by a sequence of chords. The transcription of music in a sequence of chords is a popular and active field in the domain of music information retrieval. For that matter, we should note that the MIREX[1] evaluation campaign has included an "audio chord estimation" task since 2008.

The reader must keep in mind that chord estimation is a complex and complete problem. It consequently constitutes the focus of some substantial research works. Our positioning here is radically different. We do not aim at bringing any essential contribution to the modelling aspect of the problem. Our work focuses on the use of existing chords modelling technique for the purpose of audio-identification. More specifically, we have worked with a chord estimation technique that relies on the work of Oudre [OFG11]. The approach has proved to work well, notably through its results in the latest MIREX contests. The chord estimation however stays mostly an input for the search part of our algorithm. The reasoning could thus be easily transposed to other chord estimation techniques.

We now give a brief presentation of the chord modelling problem as well as a short overview of some existing methods. The interested reader is invited to refer to more specific works if a deeper insight is desired.

17.2.2 What is a note?

In music, a note is the result of the setting in vibration of a musical instrument. When the latter is excited at a given frequency (the fundamental frequency), its physics involve the generation of vibrations whose frequencies are located at or near integer multiples of the original vibration (the harmonics) [Rig85]. However, to a human listener the resulting auditory sensation constitutes one single musical event. The perceived height of this event is called the *pitch* of the note. Psycho-acousticians define the pitch as the frequency of a pure tone (i.e. a sinusoidal waveform) that is perceived as having the same height as the note. A consequence of this definition is that we sometimes use, somewhat loosely, the terminology *frequency of a note* when actually referring to its pitch.

Two notes with pitches whose ratio is 2 are perceived as very similar by the human auditory system. This phenomenon is referred to as the periodicity of pitch-perception. Physically speaking, two such notes share a meaningful number of harmonics (for an instrument whose

[1] http://www.music-ir.org

timbre is independent of the pitch, one out of two harmonics of the lower note are common with the ones of the higher note). These notes are said to belong to the same *pitch-class* and they are given the same name $\mathcal{N}$. In order to distinguish them, we use the notation $\mathcal{N}_{i+1}$ for the note with the higher pitch and $\mathcal{N}_i$ for the other one. The interval between two such notes is called an *octave*.

The scale in use in Western music is the equal-tempered chromatic scale. Its construction process is the following. First, one reference note must be defined. Typically, in Europe we use the convention $A_4 = 440$Hz. Thanks to the octave relationship described above, we can deduce the pitches of A_i for any i. It is thus clear that the equal tempered scale only has to be defined within an octave. The principle is to divide the octave in twelve notes with geometrically spaced pitches. Mathematically, this means that the pitch of the k^{th} note is obtained by multiplying the pitch of the $(k-1)^{th}$ note by $\sqrt[12]{2}$. If we start with A_4, the names that are given to the twelve successive notes are the following: $\{A_4,\ A_4\sharp,\ B_4,\ C_5,\ C_5\sharp,\ D_5,\ D_5\sharp,\ E_5,\ F_5,\ F_5\sharp,\ G_5,\ G_5\sharp,\ A_5\}$.

17.2.3 What is a chord?

What we call a chord in music is the association of several notes. The notes of a chord can be played simultaneously or one after another (we then talk about an arpeggio). In both cases, the notes are sufficiently close in order to resonate together in the listener's ear.

The conceptualisation of such an object is of great interest for representing polyphonic music. As a result, music theorists and composers have put a meaningful effort in the description of chords construction, their classification and the ruling of their succession. For that matter, tonal harmony, which arose from the counterpoints rules that stipulate the possible arrangements between the different voices of a music, was first expressed as a theory on its own by Rameau (1683-1764). The rules formulated in this theory apply to most western compositions until now.

This theory notably takes into account the fact that the auditory sensation provided by a chord (usually called the *colour* of the chord) is rather octave independent. This means that substituting any note of the chord by another note that belongs to the same pitch-class will, to some extent, preserve the perceptual quality of the chord. As we will see, this statement is of great importance for us. It indeed means that estimating a chord does not require the estimation of the pitches of the notes involved: it suffices to determine their pitch-classes.

Of course, the rules of tonal harmony have evolved throughout history along with the music styles. The theory proposes to reference all chords that are made of a superposition of thirds (minor or major). At the time of the elaboration of the theory, the third was indeed imposing as a consonant interval in Western music. This gives rise to the triads (root note

+ the third + the fifth), the tetrads (root note + the third + the fifth + the seventh) and pentads (root note + the third + the fifth + the seventh + the ninth). As for the evolution of music styles, we can note that romantic music allowed a more generous and more flexible use of tetrads chords than classical music. Romantic composers also tended to frequently use pentads whereas they are hardly found in classical pieces. In jazz, composers have elaborated a new chords notation system. The latter is much more flexible and allows the formulation of notes combinations that are not admitted in the classical theory.

Western music strongly relies on the rules of harmony. Indeed, what music composers do is essentially designing a chords sequence (or chords progression) that fits a melody and that follows the rules in use in the chosen music style. This explains why chords are a very effective way of representing music and why there has always been a chords nomenclature throughout the ages. This truth also transposes to the listener's side. People are indeed sensitive to the notion of harmony. In other words, if one listens to two different melodies that are played on the same chords progression, the listener will find that the musics present a meaningful similarity.

17.2.4 Interest of chord estimation for audio-identification

In the domain of automatic transcription, the ultimate goal would be the development of an algorithm that can estimate all the notes played by an ensemble of different instruments. This problem, that is actively studied by the community, is quite challenging. One of the difficulties comes from the fact that it is not straightforward in the common spectral representations to distinguish the fundamental frequencies of the notes played from the associated harmonics. If several notes are played together and the harmonics of the different notes merge, we understand that the equation starts being very hard to invert.

One way to get round this problem is to simplify it. We can thus ask ourselves: "isn't there an intermediate representation that does not need the separate estimation of all notes pitches but that can still bring some meaningful musical information ?". The answer is "yes" and the intermediate information that we can target is the harmonic content. The latter is easier to characterise since it does not require the estimation of the set of notes played together but simply the overall perceived pitch-classes. This problem is tackled thanks to the chromagram and pitch-class profile representations. The chord estimation techniques usually start from these representations and aim at reconstructing the original chords sequence from the composer. In a way, they can be seen as a quantification of the information contained in the chromagram.

The satisfying thing with these estimations of the harmonic content is that they have a true perceptual correspondence. As stated before, listeners are indeed very sensitive to the

harmonic progression of music. Having available techniques that can provide an estimate of the harmony therefore presents a real applicative interest, notably in the field of audio identification.

17.2.5 Chord estimation techniques

As mentioned, chord estimation has been an active field of research, which has received numerous contributions. The methods have a common starting point that is the use of a feature that estimates the harmonic content of the signal. Although there are small variations in the way to compute it, the principle stays the same: it is a spectral representation that gives the amount of energy for any pitch-class that falls within the scope of the equal-tempered scale. Different names have been given to these representations but the most general terminology is probably *chroma* vectors.

What the methods then do is essentially looking, for each chroma vector of the analysed signal, for its best match among a set of pre-defined templates. Each template is meant to be the best chroma representation of one of the chords defined in *tonal harmony*. The templates can be either pre-defined [Fuj99, HS05, OFG11] or learnt by training [SE03]. Some methods introduce some musical knowledge in their models in order to enhance the chord estimation. In [BPP05] the succession rules of the chords are taken into account whereas some approaches estimate the overall key (i.e. the tonality) of the signal [Sai06]. Other works rely on the extraction of rhythmic information so that the chord estimation depends on the chroma's position in the bar [PP08]. Finally, some recent works propose the joint use of multiple information (key, rhythm, bass) in order to improve the chord estimation [MD10].

17.3 Chord Model

17.3.1 Introduction

We now detail the model that we use to transcribe an audio signal in a sequence of chords. More specifically, our strategy relies on the work of Oudre [OFG11], which can be described in two steps. In a first section, we give a detailed description of the computation of the chromagram. In a second section, we present the vector quantisation scheme that Oudre proposes for the transcription of a chromagram in a sequence of chords.

17.3.2 CQT & Chromagram

The starting point of a chord transcription method is the use of a spectral representation of the signal. For the same reasons as the ones exposed in section 12.2, the use of a Constant-Q-Transform (CQT)[Bro91, BP92] imposes. The chord and chromagram estimation are indeed strongly tied to the notion of note. Let us recall that the CQT is a spectral estimation tool that gives the amount of energy in the signal in frequency bands that follow the same spacing as the notes of the Western scale. Provided an appropriate choice of the parameters in the CQT computation, the frequency bands can be aligned with the frequencies of the notes.

The k^{th} bin of the Constant-Q-Transform of an input signal x is calculated as:

$$X_{(k)}^{CQT} = \Big| \sum_{n=1}^{N(k)} w(n,k)x(n)e^{-2j\pi f_k n} \Big| \quad \text{with } f_k = 2^{\frac{k}{12r}} f_0$$

$N(k)$ is the length of the window $w(.,k)$, which is proportional to the inverse of frequency f_k. In other words, as f_k increases the resolution of the transform decreases, as it happens in the human auditory system. The variable r corresponds to the number of bins per note: if we set r to 1, there are twelve geometrically spaced bands between f_k and $2f_k$. This exactly matches the definition of the chromatic equal tempered scale. Provided we choose a starting point f_0 that matches the pitch of one of the standard notes, the set of frequencies $\{f_k\}$ will be a perfect mapping of the set of pitches of the notes of the Western scale. We thus have exactly one energy band per note.

In our implementations, we use $r = 3$. This means that we have three bins that match the extent of one note of the scale. We have chosen f_1 that matches the pitch of $G_1\sharp$ so that the three energy bands with centre frequencies $\{f_0, f_1, f_2\}$ match $G_1\sharp$, the ones with $\{f_4, f_5, f_6\}$ match A_1 and so on...

The chroma vector is computed out of the spectrum obtained thanks to the Constant Q Transform. The chroma aims at reflecting the perceived pitch-classes. Its domain of definition is thus one octave, which implies that it contains $12r$ bins. One chroma bin is obtained by summing all the bins of the CQT that belong to the same pitch-class [Fuj99]. Mathematically, this gives the following formula, for $j \in \{1, 2...12r\}$:

$$C(j) = \sum_{k=0}^{k_{max}} X_{(j+12rk)}^{CQT}$$

where $k_{max} = \sup\{n \in \mathbf{N} \ / \ 12 + 12rn \leq f_{max}\}$. In simpler words, k_{max} is the total number of entire octaves in the Constant Q spectrum. We can observe that, since the chroma vector is an object that builds upon the periodicity of pitch-perception (see section 17.2.2), it is

cyclic: for any pitch-class represented by bin $i \in \{1, .., 12r\}$, the next pitch-class is given by bin $i + 1 \pmod{12r}$.

The concatenation of successive chroma vectors gives the chromagram. In order to obtain the next chroma vector, the windows w from the CQT calculation are temporally translated by a time quantity t_{hop}. In the end, the chromagram is a function of the pitch-class j and the time t: $C(j, t)$.

Oudre suggests, similarly to Bello & Pickens [BPP05], a further step in the calculation of the chromagram meant to compensate the possible detuning of a music. Given the frequency resolution of the Constant Q Transform, we can see that the proposed chroma vector contains r bins per note-class. In other words, one note and any of its octaves $\mathcal{N}_i$ is represented by the chroma bins $\{k_l...k_{l+r-1}\}$. Let us imagine that the analysed signal consists of the performance of one note $\mathcal{N}_i$, played steadily, precisely in tune. The energy should then be focused in one single energy bin k_ε. Under the conditions of a coherent tuning between the performance of the musicians and the algorithm, we would expect:

$$k_\varepsilon = \underset{j \in \{l,..,l+r-1\}}{\text{median}} \{k_j\} \tag{17.1}$$

However it happens that the tuning of the music does not correspond to the standard tuning. First, musicians sometimes use a different tuning reference than the one we described. Besides, even if the same tuning reference is used, it happens that the analysed signal has undergone some specific post-processing that infers a shift in the pitches. In order to re-establish a coherence between the actual tuning of the signal and the mathematical calculation, the idea is to circularly shift the bins of the chroma until Equation (17.1) holds.

Bello & Pickens propose the use of a peak-picking technique in order to locate k_ε. We do propose a simpler method in order to achieve the same objective. For $i \in \{1, .., r\}$, we define the probability of having the i^{th} bin corresponding to one actual note-class of the signal by:

$$P(k_\varepsilon = k_i) = \frac{\sum_{t=0}^{t_{max}} \sum_{k=0}^{11} C(kr + i,\, t)}{\sum_{t=0}^{t_{max}} \sum_{k=1}^{36} C(k, t)}$$

The probability is obtained by summing for all the different note-classes the energy that they have in their i^{th} bin (sum over k). This principle is temporally extented by summing the contributions of the successive chroma vectors (sum over t). The quantity is further divided by the total energy so that the probabilities add up to 1.

The algorithm computes the probabilities $P(k_\varepsilon = k_i)$ for all the k_i's. Finally, the situation with the highest probability $P(k_\varepsilon = k_{i_0})$ is considered to be the actual one. The chromagram is then circularly shifted so that the bin k_{i_0} becomes the median bin of the note-class.

The representation is finally brought back to a 12 bins -corresponding respectively to the 12 notes of the scale- spectral representation. This is done by summing all the bins that belong to a same note-class. This can be done either with a uniform weighting within the class, or by using a weight function that favours the median bin.

17.3.3 Chord estimation

As described before, our chord estimation algorithm is based on the work of Oudre [OFG11]. The method relies on the definition of a chord dictionary. The later contains λ *templates* $\{W_i\}_{i\in\{1,..,\lambda\}}$ that correspond to the output vocabulary of the algorithm. In other words, for any input signal, the method outputs a sequence of chords that all belong to the templates.

The method associates to each temporal frame of the chromagram $C(.,t_0)$ (i.e. to each chroma vector) a template from the dictionary. For each template W_i, the distance $d(C(.,t_0)\,;\,s_{i,t_0}W_i)$ is computed. The scale parameter s_{i,t_0} accounts for the change of amplitude between the template and the actual performance of the chord. The template realising the minimum distance is retained. In [OFG11], it is underlined that what is defined as a distance can also be seen as a probability $p(C(.,t_0)\,|\,s_{i,t_0},W_i)$. This change of modelling gives more flexibility in the matching and notably allows the taking into account of additional global statistics. If the mentioned statistics is denoted by α, the estimated probability becomes $p(C(.,t_0)\,|\,\alpha,s_{i,t_0},W_i)$.

As far as α is concerned, Oudre's suggestion is to bring into play the probability of occurrence of one chord in the signal. In practice, if we denote by α_i the probability of occurrence of template W_i, we define:

$$p(C(.,t_0)\,|\,\alpha,s_{i,t_0},W_i)=\alpha_i\,p(C(.,t_0)\,|\,s_{i,t_0},W_i)$$

For the computation of $p(C(.,t_0)\,|\,s_i(t_0),W_i)$, several statistical observation models are proposed by Oudre. The two models that we exploit here are the following:

$$p_{gauss}(C(.,t_0)\,|\,s_{i,t_0},W_i)= \exp\left[-\left(\frac{1}{2\sigma^2}d^2_{Eucl}(C(.,t_0)\,;\,s_{i,t_0}\,W_i)+\frac{M}{2}\Big(\log(2\pi)+\log(\sigma^2)\Big)\right)\right] \tag{17.2}$$

$$p_{gamma}(C(.,t_0)\,|\,s_{i,t_0},W_i)=\exp\left[-\left(\beta d_{IS}(C(.,t_0)\,;\,s_{i,t_0}\,W_i)+ M\Big(\log(\Gamma(\beta))-\beta\log(\beta)+\beta\Big)+\sum_{k=1}^{M}\log\big(C(k,t_0)\big)\right)\right] \tag{17.3}$$

where M is the number of bins of the chroma vector, σ and β are parameters of the models that allow to tune the influence of α on the output of the algorithm (see [OFG11] for details on their tuning). The notations d_{Eucl} and d_{IS} respectively stand for *Euclidean distance* and *Itakura-Saito distance*.

The scale parameter is fixed to the value that minimises the distance between $s_{i,t_0}W_i$ and $C(.,t_0)$. It can be analytically computed by zeroing the derivative of the above quantities with respect to s_{i,t_0}. The obtained expressions are the following:

$$s_{i,t_0}^{gauss} = \frac{\left(C(.,t_0) \mid W_i\right)}{\left(\|W_i\|_2\right)^2} \tag{17.4}$$

$$s_{i,t_0}^{gamma} = \frac{1}{M} C(.,t_0) \oslash W \tag{17.5}$$

where $(.|.)$ denotes the scalar product, $\|.\|_2$ the Euclidean norm and $\oslash$ the term-wise division.

In order to jointly estimate $p(C(.,t_0) \,|\, \alpha, s, W_i)$ and α, an iterative process is adopted:

`// Init of` α `with a uniform law`
$\forall\, i \in \{1,..,\lambda\}\ \alpha_i \leftarrow \frac{1}{\lambda}$;
`// Main loop`
while α *has not converged* **do**
  $\forall\, i \in \{1,..,\lambda\}\ \alpha_i \leftarrow \left\langle \frac{\alpha_i p\left(C(.,t) \,\middle|\, s_{i,t}, W_i\right)}{\sum_{k=1}^{\lambda} \alpha_k p\left(C(.,t) \,\middle|\, s_{k,t}, W_k\right)} \right\rangle_t$;
end

The probability α_i of occurrence of one chord W_i in the input signal C is estimated as the temporal average (denoted by $\langle . \rangle_t$) of the contributions of W_i in the generation of $C(.,t)$ taking into account the current values of α. It is shown in [OFG11] that this iterative algorithm can be directly derived from the theoretical expression of the Expectation Maximization algorithm [DLR77].

Since we are trying to build a perceptually meaningful fingerprint out of this representation, we do not need to keep as much information as the current output. Chords are indeed slowly varying in most music types and the listener cannot perceive fast variations in the harmony. This justifies the idea of a downsampling of the chords rate. Besides, this will allow to gain compactness in the fingerprint and ease the search step. A reasonable order of magnitude for the chords rate seems to be around 1s. In our implementation, the chords rate is taken equal to 0.8s.

17.4 Distance between chords sequences

17.4.1 Introduction

Given the chord transcription model described above, we now consider that any two music signals are represented by two chords sequences. In the context of audio-identification, we must set up a method for determining, on the basis of the chords transcriptions, whether the two signals are musically similar. We consequently propose the definition of a distance in the space of the chords sequences.

Our definition of the distance relies on the following considerations. If we have a full match between both chords transcriptions we will naturally consider that the signals are similar. But it may happen that two music signals that should be considered as similar are transcribed by slightly different chords sequences. Two factors account for this. First, the estimation of chords is not extremely robust to the distortions. The typical example is the confusion between one major triad and its minor equivalent. The only difference between these two is the third (there is a semi-tone between the respective thirds of the two chords). If one imagine a bit of music where a chord is actually played without the third it is impossible to determine whether we are dealing with the major or minor chord. However, the algorithm has to choose one chord among the templates. As a result, the third that has the most residual energy (depending on the ambient noise, the reverberation, the mixing of the other notes harmonics) will trigger the choice of one of the two chords. However, this choice, based on residual energy, may change when the signal is distorted. The application of equalisation, for example, may result in the predominance of the other third and consequently the choice of the other chord. Secondly, the idea of this work on approximate matching is to allow a fair amount of distortion between two performances of the same item. On the one hand, one can think that part of these distortions will be absorbed in the chords modelling. If two musics are similar, they should indeed share the same chords progression and be transcribed in the same way. However, it may happen that from one version to the other, some particular chords are slightly changed (typically in the bridges) thus resulting in a different transcription. If both performances are not at the same tempo, it may also happen that one of the transcription has one or more additional chords in its transcription (thus making the signal longer).

Under the light of these considerations, the necessity of setting up a distance that relies on an approximate matching strategy emerges. Dynamic programming [Bel54] is a technique that allows to simultaneously take into account the changes that can occur in the estimation of one precise chord (substitution) and the presence of additional chords that are not in the initial transcription (insertions, deletions). We have consequently adopted a dynamic programming scheme for the calculation of the distance between two chords sequences. The

distance computation is detailed in the following section.

17.4.2 Dynamic programming

Let us consider two chords sequences $W_{i_1}...W_{i_K}$ and $W_{j_1}...W_{j_L}$. In order to determine the distance between these chords sequences, we compute their dynamic alignment. When dynamically aligning the two sequences, one has to define three types of multiplicative penalties.

- $f^d(W_{i_k})$: penalty assigned to the deletion of chord W_{i_k}
- $f^i(W_{i_k})$: penalty assigned to the insertion of chord W_{i_k}
- $f^s(W_{i_k}, W_{j_l})$: penalty assigned to the substitution of chord W_{i_k} by W_{j_l}.

In our implementation f^d and f^i are both taken constant and equal to 0.07. Taking advantage of the fact that the template chords live in an Euclidean space, we define f^s by:

$$f^s(W_{i_k}, W_{j_l}) = \cos(W_{i_k}, W_{j_l})$$

With such a penalty, $f^s(W_{i_k}, W_{j_l}) = 1$ if W_{i_k} and W_{j_l} are colinear and $f^s(W_{i_k}, W_{j_l}) = 0$ if they are orthogonal.

Dynamic programming consists of iteratively filling a scoring matrix D. For any $(k, l) \in \{1..K\} \times \{1...L\}$, $D(k, l)$ contains the score of the alignment of the subsequence $\{W_{i_1}...W_{i_k}\}$ with the subsequence $\{W_{j_1}...W_{j_k}\}$. D is computed in the following manner:

$$D(k,l) = \max \left\{ \begin{array}{l} D(k, l-1).f^d(W_{j_l}) \\ D(k-1, l-1).f^s(W_{i_k}, W_{j_l}) \\ D(k-1, l).f^i(W_{i_k}) \end{array} \right\}$$

The score of the alignment of $\{W_{i_1}...W_{i_K}\}$ with $\{W_{j_1}...W_{j_L}\}$ is finally given by $D(K, L)$. Figure 17.1 shows an example of matrix D.

As far as the implementation is concerned, it is strongly advised to work with the logarithm of the quantities detailed above. In this way, the multiplications are turned into additions and the numerical range of the processed numbers radically diminishes. This prevents the risk of overflow of the numerical representations that can occur when multiplying small numbers. Besides, the quantity $-\log(D(K, L))$ can be considered as the distance between the two chords sequences.

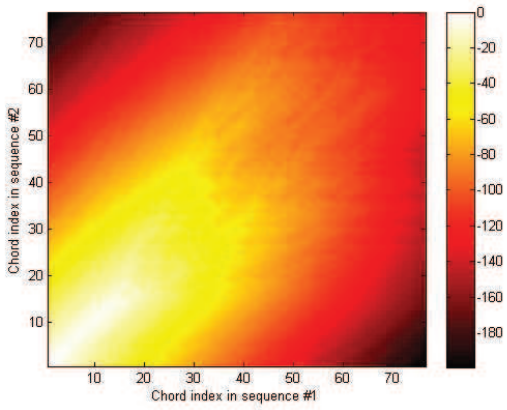


Figure 17.1: Example of scoring matrix when dynamically aligning two chords sequences. The values of the matrix are figured by their colours, according to the side colourbar. The implementation of the alignment is done in an additive fashion with the logarithms of the penalties, hence the negative values.

17.5 Fast Search Strategy

17.5.1 Introduction

17.5.1.1 Runtime considerations

The distance computation described above constitutes a relevant theoretical basis for the comparison of two signals represented by their chords sequence. However, it is quite costly in terms of CPU. The number of operations required to compare two sequences of size K and L is indeed $O(KL)$. As a consequence, when the system has to identify an unknown signal, it would be unreasonable to compute its distance with all the references of the database. Such a system would indeed be hard to make scalable. We have thus elaborated a fast search strategy that is meant to quickly select in the database the references that are the closest to the unknown signal.

17.5.1.2 Methods from the approximate string matching domain

In the domain of approximate string matching, a number of approaches have been developed in order to perform efficient string comparisons. These methods notably target the search of a given input string (or sequence) in a database containing a large set of reference strings, with possible distortions. As these specifications match our use-case, it is worth studying the state of the art of the domain. We give here a brief overview of the common techniques of the field.

First, we have the online strategies [WF74, Sel80, BYN96]. These do not rely on a pre-processing of the text and use various heuristic in order to parse the text as fast as possible. Some of them notably exploit the redundancy that exists when the database is scanned with a large sliding window that moves by only one character at each step [LMSS95, KP00]. However, for the reasons exposed in section 8.1.4, these methods cannot compete with index-based strategies when the database becomes large.

The index-based methods start with a clear advantage in terms of speed but they face the exactness problem that we mentioned in section 8.1.4. Indexes are inherently unadapted to approximate queries. The methods from the domain consequently propose various clever workarounds in order to make use of indexes in the context of approximate string matching. The article [NByST00] notably groups them in three families.

The first type of methods [JU91, Cob95] generates a neighbourhood of the searched sequence. More precisely, all sequences that lie at a distance smaller than a maximum distortion threshold from the query are generated. The idea is that if the input sequence has been slightly distorted compared with the corresponding reference, the undistorted version should be found in the neighbourhood. The index is then queried with all the sequences from the neighbourhood. If the undistorted sequence is actually in the neighbourhood, the corresponding reference will be returned by the index.

The second type of methods [NBy98, Shi96] relies on partitioning. These methods also hypothesise a maximum level of distortion on the input sequence. The idea here is that if the input sequence is cut in 'small' subsequences, then a significant number of these subsequences will be undistorted. Constraining the maximum number of distortions indeed involves that only some of the subsequences will be hit by the distortions. The index is then queried with all the different subsequences. Aggregating the index outputs obtained with all the subsequences allows to determine the reference that most likely matches the entire input sequence.

The third group of methods [Mye94, NBY00] use both described strategies. They typically split the query in subsequences and generate the neighbourhood of these subsequences. According to [NByST00], the methods from the third group are the most promising in terms of efficiency. The method "Basic Local Alignment Search Tool" [AGM$^+$90], which has been developed in the context of bio-informatics (on typical issues such as retrieving a slightly modified sequence of ADN in large database), belongs to the third group. It has become a quite popular method for fast search and has been used in various fields. We have consequently led some preliminary experiments in adapting the method to our use-case. These showed that the method was successfully meeting our needs in terms of precision and efficiency. As a result, we made the choice to integrate this search method in our final algorithm.

17.5.1.3 Focus on the BLAST strategy

The principle of BLAST is to extract subsequences of a given length w from the query signal. For each extracted subsequence, the algorithm generates its neighbourhood. All the resulting subsequences are used to query an index. The reference database has been preprocessed beforehand so that the index contains the subsequences of length w from the references. By aggregating the index outputs obtained for each neighbouring subsequence, we can score each reference with respect to the number of times that they were hit. It is however to be noted that this BLAST score is not a straightforward function of the dynamic-programming distance we described above. BLAST is mostly a heuristic that allows to target a set of candidates among which the closest reference to the query (in the sense of the dynamic-programming distance) will hopefully be. The success of the method notably depends on the tuning of the parameters.

17.5.1.4 Description of our search strategy

In the following, we detail our implementation of BLAST's principles. The presentation is organised as follows. At first, we move back to our stream-based use-case: we detail the framing that we apply to the input stream. Each frame is then a query signal for the fast search method. We then detail how we extract the subsequences of length w from each query. This is followed by a precise description of the neighbourhood generation that is applied to each subsequence. We then explain how we gather the index outputs obtained with each subsequence, which gives a BLAST score for each reference. Since the score is not directly linked to the distance we elaborated, our search method is completed by a fine matching step, which consists of computing the distance between the references with the best BLAST scores and the query.

17.5.2 Framing of the broadcast

The work conducted in this chapter significantly differs from the one described in the chapter about *exact matching* for the following reason. As we can see, the representation of the signal is now (hopefully) representative only at a wide scale. It is indeed clear that we will not be able to uniquely identify a reference thanks to one single chord or a a succession of two chords. In order to spot a reference in the broadcast's transcription in chords, we must look for the entire set of chords composing the reference.

The framing of the broadcast must be tuned accordingly. Given the length of the references that we use (60s) and the chords rate of our transcription method (1 chord per 0.8s), the references are transcribed by sequences of 76 chords. The minimum length of a frame

is thus 76 chords. In order to limit the number of searches performed, it is interesting to manage larger bits of broadcast at once. The drawback, in return, is that increasing the frame length increases the noise in the search. Indeed, the presence of chords that do not belong to the target reference will generate irrelevant search queries. This particularly becomes true if another reference is included within the same frame. The choice of the length of the frames must be done according to these considerations. Besides, in order to make sure that any reference is fully included in at least one search frame, these must have an overlap of 76 chords at least. In our implementation we have used a frame length of $L_f = 240$ and a hop size of $h_f = 240 - 76 = 164$.

17.5.3 Extraction of subsequences

According to BLAST's strategy, when analysing an unknown signal represented by its chords sequence, the algorithm extracts subsequences of length w. In our implementation we have selected a length $w = 4$, which has appeared in our experiments as a good compromise between computation time and precision. The experiments conducted by the authors of [AGM+90] have led them to the same choice for w in the bioinformatics context. Let us note that the extraction of subsequences that we propose is asymmetric. In the unknown signal, we extract subsequences of length w with no overlap (the hop is w), whereas the subsequences of the references have been extracted with a unitary hop. We can note that the presence of the unitary hop (on one side or the other) is necessary when using search frames that do not have a unitary hop (such as ours). It indeed makes the index robust to the possible shifts of the start index of the broadcast reference within the search frame. We actually find it preferable to increase the size of the database and its construction time, which is offline, rather than increase the number of query subsequences. The processing of the latter indeed involves several steps (neighbourhood generation and query to the database) whose computational cost is meaningful.

17.5.4 Generation of the neighbourhood

For each extracted subsequence, the algorithm generates its neighbourhood. For a given threshold T_{Neigh}, the neighbourhood is the set of subsequences whose distance to the extracted subsequence is smaller than T_{Neigh}. Let us note that the index suggested in BLAST only contains the subsequences from the database of length w. For that matter, it seems delicate to take into account the insertions and deletions when generating the neighborhood of the searched subsequences. The latter would indeed generate subsequences with a different length, which could not be used to query the index. We consequently restrict our construc-

tion of the neighbourhood: only subsequences obtained by substitutions are considered at this stage of the search. Besides the fact that this seems imposed by the fixed-length index, this presents the further advantage of limiting the computations. Building the neighbourhood has indeed a meaningful cost in the total computation time and enlarging the notion of neighbourhood obviously increases this cost.

For one given subsequence $W_{s_1}...W_{s_w}$, we propose a computation of its neighbourhood relying on a tree[2] (see Figure 17.2). The tree we use is the one generating all subsequences of length w. The first layer contains λ nodes corresponding to the λ possible chords. In the node corresponding to chord W_j we store the distance $d(W_j\,;\,W_{s_1})$. The distances between chords can advantageously be stored in a matrix so that the algorithm does not have to compute them each time. The second layer corresponds to the possible 2 chords sequences and therefore contains λ^2 nodes. Naturally, each node of the first layer is connected to λ nodes of the second layer corresponding to the λ different chords. In the node of the second layer corresponding to chord W_l and that is reached by traversing the tree through the 1st layer's node corresponding to W_j, we store the distance $d(W_jW_k\,;\,W_{s_1}W_{s_2})$. The distance calculation here does not include the insertions and deletions. It is thus simply obtained by:

$$d(W_jW_k\,;\,W_{s_1}W_{s_2}) = d(W_j\,;\,W_{s_1}) + d(W_k\,;\,W_{s_2})$$

which, in terms of node, gives:

$$N_{2,k} = \text{parent}(N_{2,k}) + d(W_k; W_{s_2})$$

By iterating the process, we obtain a tree with λ^w leaves that correspond to all possible subsequences of length w and that contain the distances between these subsequences and the query subsequence. An interesting feature of the tree is that we can prune the branches as soon as one of their nodes contains a distance above the treshold. It is indeed clear that the distances are monotonic and increasing when traversing the tree. This allows to save a substantial computation time.

17.5.5 Aggregation of the database outputs

For each subsequence extracted from the analysed signal, its neighbourhood is generated. The algorithm gathers all these neighbourhoods which gives a list of subsequences. The index is subsequently queried with these subsequences. Each query gives rise to an output including: the references containing the subsequence and the index of this subsequence in

[2]For this reason, we did switch this part of the code in C. The imbricated loops involved are indeed very badly managed by Matlab.

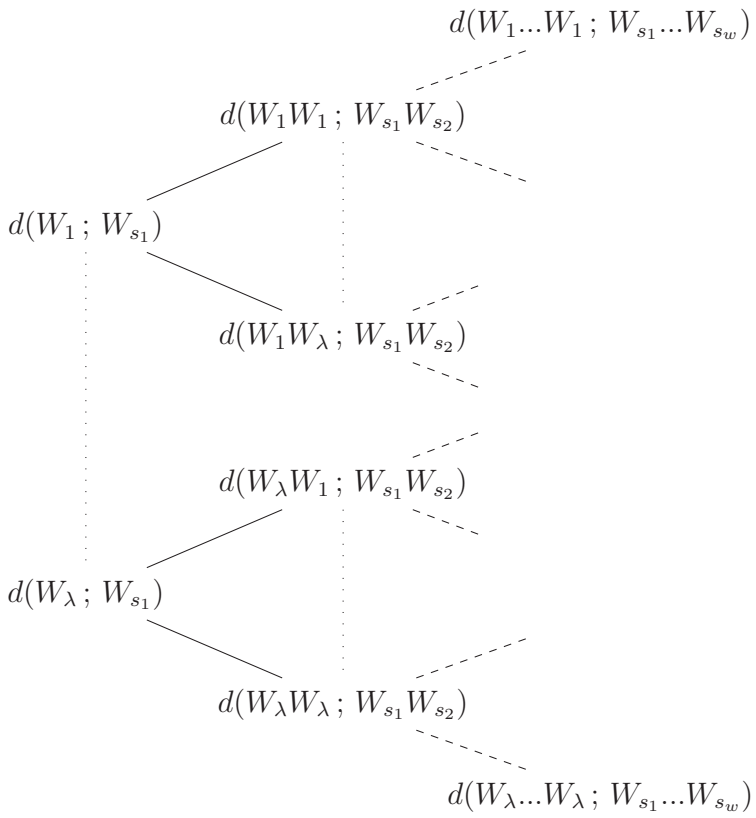


Figure 17.2: Illustration of the tree used for the neighbourhood generation. The leaves of the tree contain the distance of any subsequence of length w to the extracted subsequence $W_{s_1}...W_{s_w}$.

the references' overall chords sequences. We propose here an additional aggregation step of these index outputs that is not described in the original paper [AGM+90]. The idea is that we will not only look for references having a lot of subsequences in common with the list but we also expect the occurrences of these subsequences in the unknown signal to be correlated with their occurrences in the corresponding reference.

The strategy that we consequently suggest is the following. For each reference hit by the index, we build a histogram. When for a given subsequence located at position i in the unknown signal, the index returns a reference containing the same subsequence at position j, we update the concerned histogram by incrementing the bin containing the value $i - j$. However, we must recall that some of the query subsequences are actually extracted from the unknown signal whereas others are simply in the neighbourhood. In order to take this into account, the increment is a function of the distance between the actually extracted subsequence and the query subsequence. Typically, one can go back to the penalty functions presented in section 17.4 by taking an increment equal to e^{-d}, where d is the distance between the two subsequences.

For all subsequent hits concerning this reference, we fill the histogram with the corresponding increments in the corresponding index shifts bins. At the end of the processing, we thus have as many histograms as hit references. In order to locate the 'good' candidates for a match, we set a simple threshold mechanism on the highest peaks of the histograms. All references having a histogram maximum above T_{hist} are potential matches to the unknown signal. The argument of the maximum in the histogram gives the start index of the reference in the unknown signal.

It is interesting to note that when designing the histograms, we have to determine the width δ_{hist} of the bins. The index-shifts being integers, it seems natural to think of $\delta_{hist} = 1$. However, if one wishes to increase the robustness of the index-based search step towards insertions and deletions, one can play on δ_{hist}. For instance, setting $\delta_{hist} = 2$ authorises one uncompensated insertion or deletion. In our experiments, the necessity of loosening the search in this way has not been obvious. We have therefore kept working with $\delta_{hist} = 1$.

17.5.6 Fine matching of the candidates

The previous step outputs a set of candidates. These have obtained a score above a certain threshold T_{hist} in the BLAST search. However we must keep in mind that this search step is essentially a heuristic and that the BLAST score is not a straightforward function of the distance. We thus propose a finer matching step which consists of calculating the actual distance between the candidates and the concerned broadcast's sections. This is done thanks to the methodology described in section 17.4. As mentionned, this calculation is CPU

demanding. However, it is now only performed with a reduced set of candidates.

The algorithm finishes with a detection step, based on a post-processing of the calculated distance. This can consist of a simple threshold on the distance or include some kind of temporal integration (forbid the presence of two simultaneous different references, forbid the presence of another reference in the close neighbordhood of a firstly detected one...).

17.6 Experiments and Results

17.6.1 Experimental framework

Our experiments rely on the framework described in section 9.3.2. Let us recall that the streams we use for the evaluation of exact matching audio-fingerprint methods consist of real radio broadcasts. As such, these occasionally feature broadcasts of live or acoustic versions of some titles. Provided that we can put together a database of references that includes the studio versions of the mentioned titles, we are in position to define an identification use-case which lies in the field of approximate matching. This, of course, also requires the adaptation of the manual annotations so that they include the XML blocks corresponding to the broadcasts of the live/acoustic titles.

On the whole, and taking into account the dataset that we have available, putting together such a corpus has required a meaningful and specific time investment. This explains why we had to restrict ourselves to a middle-size experiment. More precisely, as far as the analysed stream is concerned, we have worked with 24h of the French radio RTL. These contain one program that essentially features live performances of contemporary titles. In total, the stream contains 107 annotated music titles, 99 of which are studio versions whereas the 8 remaining are live performances. We consequently gathered a references database that contains the titles that were broadcast in their studio versions plus the studio versions of the titles that were broadcast live. Let us note that the references corresponding to the titles broadcast in their studio versions (exact matches) were not captured on the same medium, which involves the presence of the post-processing distortions traditionally handled in the exact-matching use-case (see section 8.1.3). We completed the filling of the database with various references, for which we made sure that they were exclusively music signals (no speech, no advertisement, no corrupted signal). Under these conditions, we could assemble a database that contains 2400 pop-rock music titles with their corresponding identifiers.

Let us note that such a database still constitutes a realistic reference set for many applications. It is for instance noticeable that the assembled database covers all the music broadcasts of two different radio channels during two weeks. Besides, such a quantity of

references is large enough to get a fair idea of the system's capabilities. Given the sampling rate of the chord estimation method, the algorithm can output the detection of any reference every 0.8 seconds. In terms of false alarms, this means that the algorithm has the possibility to output around $\frac{24\times3600}{0.8} \times 2400 = 259,200,000$ of them. This figure confirms the relevance of the experiment.

17.6.2 Baseline experiments

The first experiment will serve as baseline. All subsequent results will be given in comparison with the ones of the baseline in order to make possible the reading of the results evolution.

The parameters used in this experiment for the chord model are the ones recommended in [OFG11]. The chords dictionary (see section 17.3) is composed of the 24 major/minor binary triads, as represented in Figure 17.3(a). The distance between a chroma vector and a chord template is evaluated thanks to the Gamma law, according to equation (17.3). In the fine matching comparison (see section 17.4) between the candidates sequences and the associated sections of the broadcast, we use the following permutation penalties: $f^s(W_i, W_j) = cos(W_i, W_j)$. The associated *penalty matrix* is shown in Figure 17.3(b).

Finally, Figure 17.3(c) shows the results of the algorithm on a classical ROC diagram. As most detection systems, our algorithm's output relies on a set of candidate detections with given scores. The candidate references are originally selected in the BLAST step and they are finally scored thanks to a dynamic alignment. In order to complete the process, one can simply set a threshold on the score. Only the candidates with a score above the threshold are output by the algorithm. In such a configuration, one can evaluate the algorithm's output with different threshold values. This is the object of the ROC curve. Each point of the curve corresponds to the results obtained by the algorithm with a specific threshold value. Such a curve thus allows to observe the overall response of the algorithm and to compare different algorithms independently of the final post-processing.

The results show that this novel method is able to go beyond the limitations of the traditional fingerprint algorithms that are confined to *exact matching*. We can indeed see that some of the points of the ROC curve have a higher Y-coordinate than the number of exact matches in the corpus. The counterpart of this enhanced flexibility of our method is a much higher number of false alarms. This is not surprising since by trying to enlarge the scope of a representation (in order to cover the different versions of one same title) we necessarily increase the risk of collision between two different titles. In the following experiments, we will see if the tuning of the parameters can lead to a reduced number of false alarms while keeping the same number of correct detections.

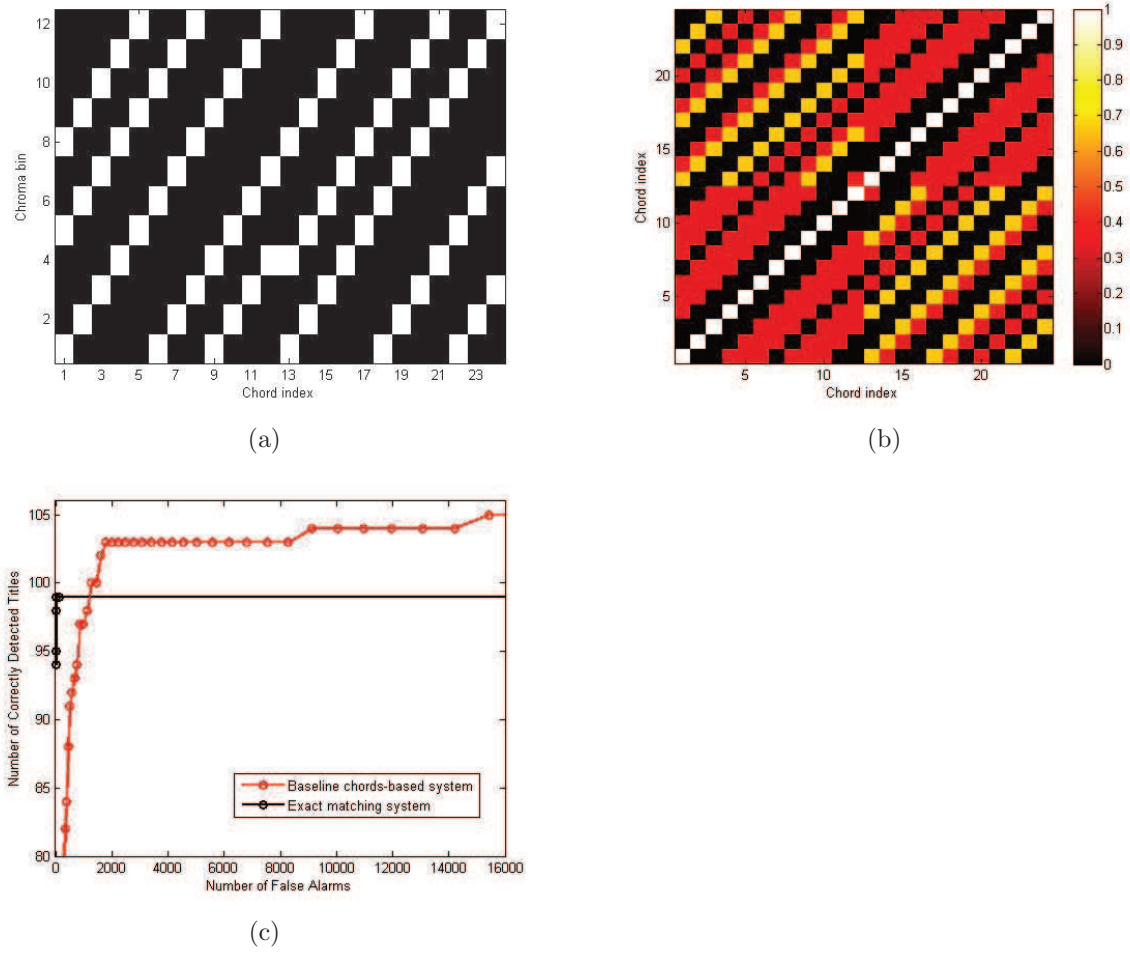


Figure 17.3: Results obtained with the chords sequence matching algorithm. The dictionary consists of the 24 major and minor triads modelled in a binary fashion (a), the penalty for substituting one chord by another is the cosine between their chroma representations as shown in the *penalty matrix* (b). The ROC diagram (c) shows the obtained detection curve in red. The typical ROC curve of an exact matching algorithm is plotted in black.

17.6.3 Influence of the substitution penalty matrix

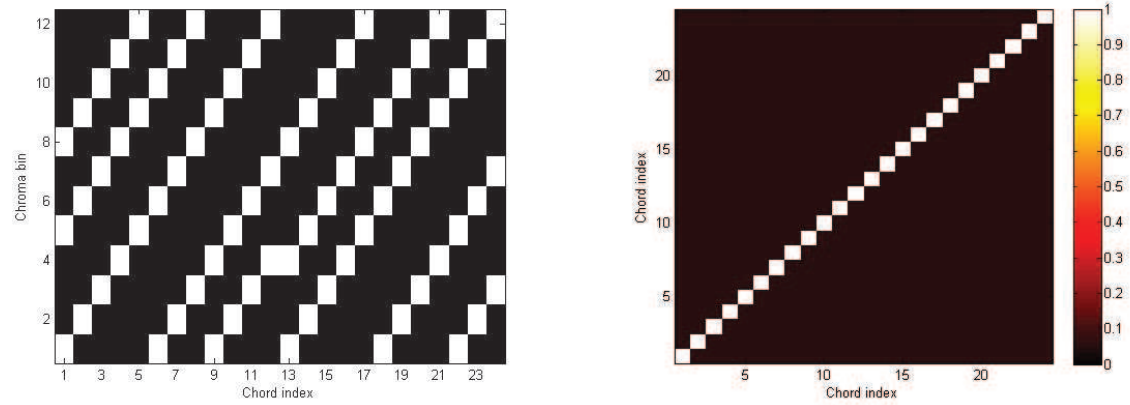
This experiment has the same parameters tuning as the first one. The difference lies in the fact that, in this experiment, the substitutions are forbidden in the final dynamic alignment. Mathematically this comes to using the Identity matrix as the penalty matrix, such as shown in Figure 17.4(b).

On the ROC curves, we can see that the baseline has a better optimal operating point (at the top left) with 103 good detections for 1794 false alarms. Conversely, we observe that for lower detection results (lower than 100 correct detections), the second method generates less false alarms. This comes from the fact that the penalty matrix now forbids the substitutions, hence limiting the possibility for the second method to match different titles that present similarities. Let us finally note that in spite of the diminishing of the number of false alarms linked to the change of penalty matrix, the chords sequence based approach is still far less precise (i.e. generates more false alarms) than the exact matching approach.

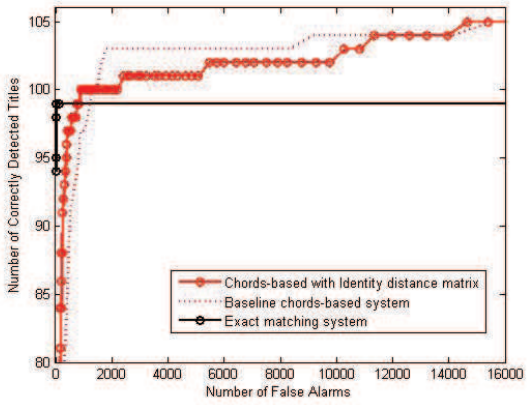
17.6.4 Influence of the probabilistic modelling

In the following experiment, we get rid of the probabilistic aspect in the chord modelling strategy. This can be done by several means, let us for instance force α to be the uniform distribution. It is remarkable that under these conditions the chords estimation algorithm becomes strictly equivalent to the first work on chords published by Oudre [OGF09]. All other parameters are kept identical to those of the baseline experiment.

Surprisingly, the results are much better with this simpler version of the chord estimation technique (see Figure 17.5(c)). There are two factors that we can consider for this evolution. First, we must keep in mind that our algorithm drastically downsamples the chord rate (we keep around one chord per second). By doing so, we are already smoothing a lot the raw output of the chord estimation proposed by Oudre. Yet, if we look at the benefits expected from the introduction of the occurrences probabilities of the chords (in the distribution α), it was mostly a smart smoothing effect on the algorithm output. Stacking the smoothing steps is usually to be avoided since it blurs too much information. That might explain the lower performance of the algorithm in which we have kept working with α. The second factor that we can consider is the difference of final purpose between Oudre's work and ours. What Oudre's algorithm tries to do is extracting from an unknown signal the chords sequence that is the closest to what a human would transcribe. In our use case, the aim is to extract from the unknown signal a chords sequence that is the closest to the one that has been extracted by the same algorithm from the corresponding (undistorted) reference. This change of paradigm may also explain why an additional feature might be good in one case and bad in the other.



(a) (b)



(c)

Figure 17.4: Results obtained with the chords sequence matching algorithm when changing the *penalty matrix*. The dictionary still consists of the 24 major and minor triads (a) but the dynamic alignment virtually forbids any chord substitution (b). The ROC diagram (c) shows the obtained detection curve in solid red while the footprint of the baseline system is in dotted red. The typical ROC curve of an exact matching algorithm is still plotted in black.

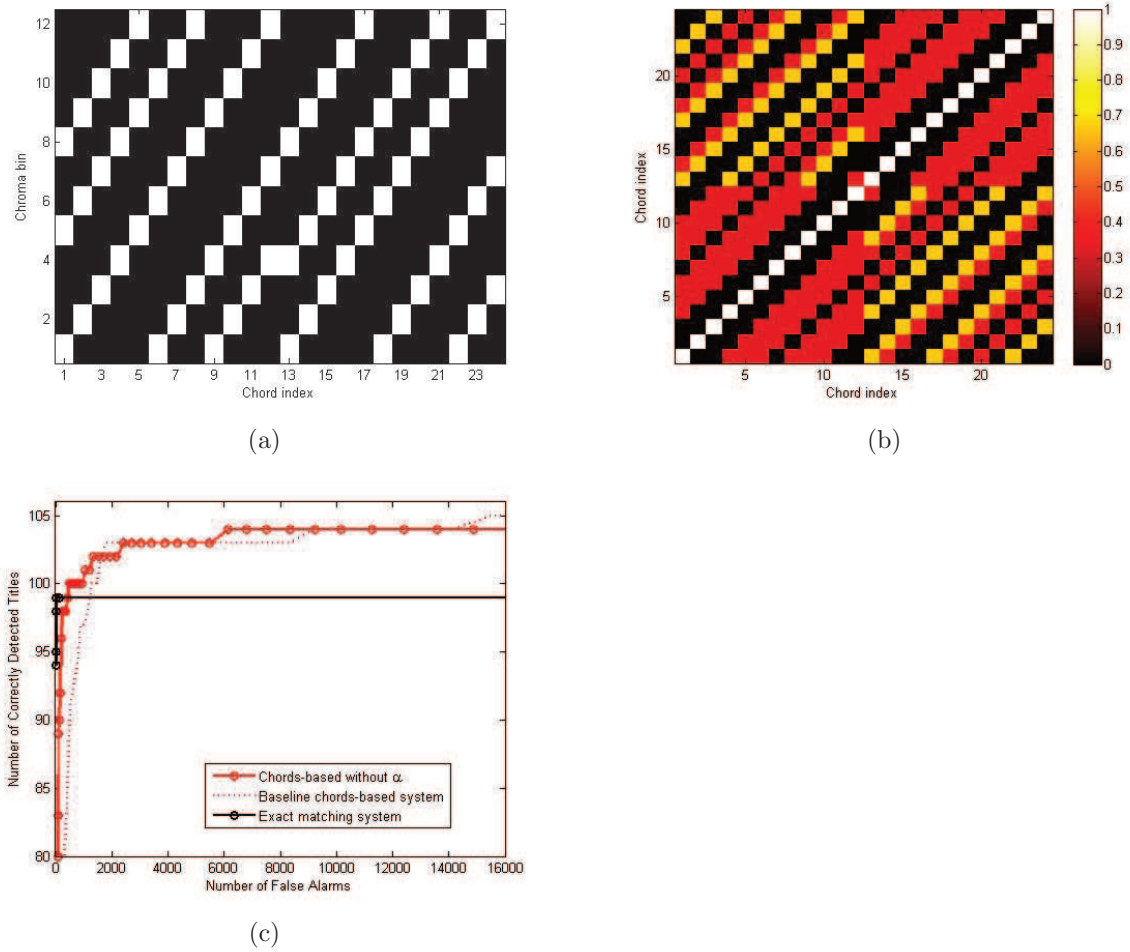
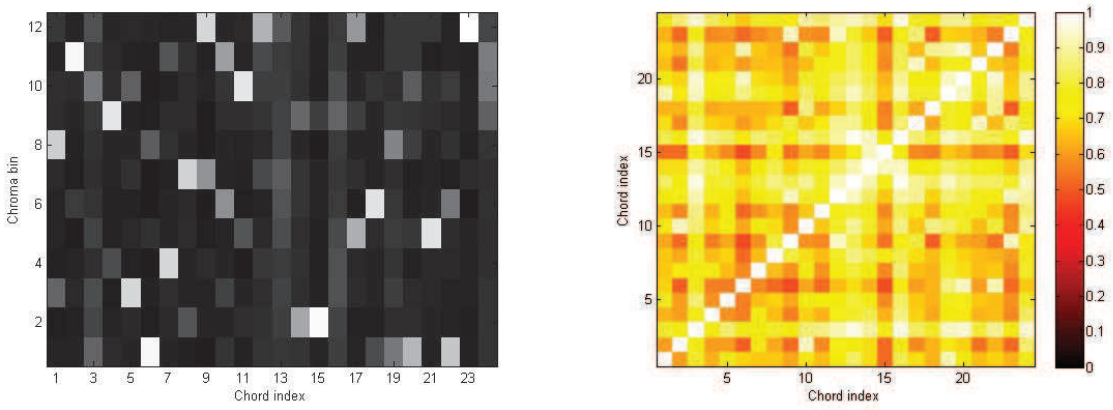


Figure 17.5: Results obtained with the chords sequence matching algorithm when removing the probabilistic aspect in the chord model. The dictionary consists of the 24 major and minor triads (a), the penalty for substituting one chord by another is the cosine between their chroma representations as shown in the *penalty matrix* (b). The ROC diagram (c) shows the obtained detection curve in solid red while the footprint of the baseline system is in dotted red. The typical ROC curve of an exact matching algorithm is plotted in black.



(a) (b)

Figure 17.6: Dictionary obtained thanks to the LBG algorithm (a) and its associated cosine penalty matrix (b).

17.6.5 Change of chords dictionary

In the following experiment, we did set up an LBG quantisation strategy [LBG80] in order to build a new dictionary of chords. At some point in our work, we did indeed wonder if the set of template chords we used was the best. In our overall processing the chord estimation is merely a way to quantise the chroma vectors. It can thus be thought that there may be a quantised family of chroma vectors that better suits our problem.

In short, LBG is a learning algorithm that clusters the vectors that it has been fed with. The output of the algorithm is the centroids of the clusters. Each centroid represents all the vectors that are part of its cluster. The LBG was run on the chroma vectors extracted from the references. In order to draw a fair comparison, we constrained the algorithm to a 24-clusters partition. We used the 24 corresponding centroids as the templates of the chord estimation dictionary.

Since it has been shown that the performance increases without α, we keep working without the EM algorithm. For mathematical convenience in the LBG computation, we work with the Euclidean distance when clustering the chroma vectors (more precisely, our clustering strategy involves the minimisation of the squared error distortion). To keep a coherence with the further algorithm steps, we use in the chord estimation process the Gaussian observation law (Equation (17.2)), which is also directly linked to the Euclidean distance.

The obtained dictionary is represented in Figure 17.6(a) while its associated cosine penalty matrix is shown in Figure 17.6(b). We do not display the associated ROC curve because the results under this configuration are actually very poor. The number of false alarms is in fact so high that the ROC curve does not fit in the graph with the previously used scales.

A careful analysis of the results led us to the conclusion that the underachievement may be linked to the weak standard deviation of the penalty matrix values. It is indeed clear, looking at the graphical representation of the penalty matrix, that the latter is far less contrasted than the previous ones. In terms of dictionary, this means that the templates are too close to each other.

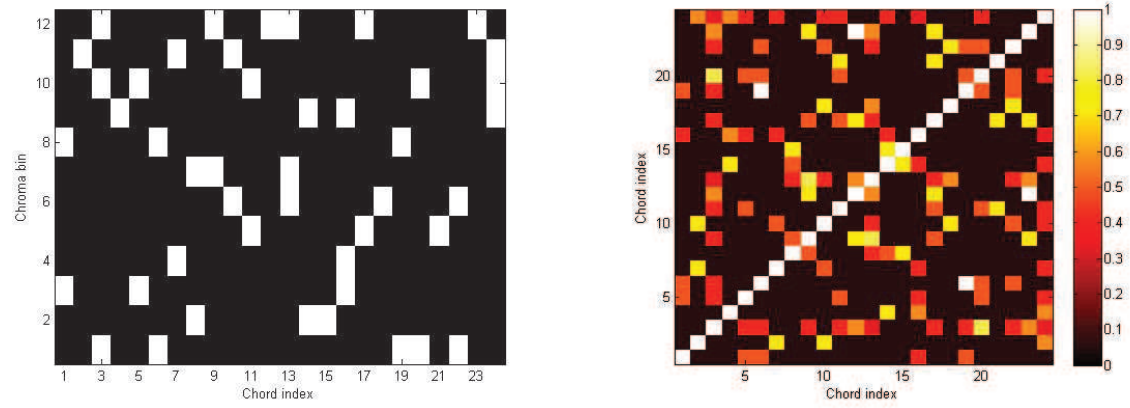
In order to test this hypothesis we try to space out the templates in the chroma vector space, while respecting the learning done with LBG. One simple way to do that is to binarise the learnt templates. This is done thanks to simple threshold mechanism. The obtained dictionary is shown in Figure 17.7(a) with its associated penalty matrix 17.7(b). The results are far better once the vectors have been made binary. We can indeed see on the ROC curve 17.7(c) that the performance is now close to the one of the baseline system.

We can of course deduce that this approach did not bring a significant improvement of the results. It is however worth having a look at the dictionary obtained thanks to the LBG learning. The LBG algorithm actually exhibits the reference chords that correspond to the minimisation of mathematical criteria. For that matter it is rather interesting to note that some of the learnt chords, after binarisation, are close to the standard triads. We notably notice that they possess an abundant number of fifths. This tends to prove that the standard chords that we used in the previous experiments, inspired by music theory, actually represent a kind of mathematical optimum. It is though noticeable that the learnt thirds are scarcer.

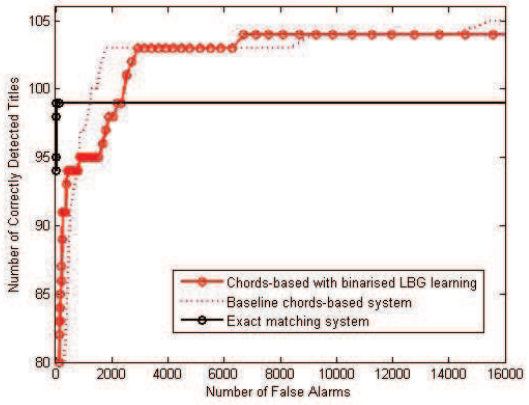
17.7 Synthesis

In this chapter, we have described an identification technique relying on the transcription of audio signals in sequences of chords associated with an indexing and an alignment strategy. This novel identification paradigm allows us to go beyond the traditional limitation of classic fingerprint algorithms that can only identify exact matches. Indeed, our experiments show that the transcription of the signal in a chords sequence is, to some extend, robust to the change of performance of a given music title. However, this ability to handle approximate matching comes at the price of a reduced precision. The presented system, due to its looseness in the representation, generates a meaningful number of false alarms.

A careful analysis of the results led us to think that we should increase the level of information that is included in the representation. The difficulty, of course, is to keep the representation loose enough for the identification of approximate matches. Since the actual information that we use deals with the harmonic content, it seemed natural to think of rhythm as an additional piece of information. We did some preliminary testing in which the chords search was used as a front-end processing and the rhythmic information was afterwards



(a) (b)



(c)

Figure 17.7: Results obtained with the templates learnt thanks to LBG and then spaced out through binarisation (a). The binarisation process leads to a penalty matrix with much more contrast (b), which in turn allows the method to reach scores that are close to the ones of the baseline system (c).

used in a “checking” step. However, in addition to being unaesthetic, the approach was not really satisfying. The correlation of pure rhythmic information that we set up tended to be computationally expensive and not really accurate. The object of the next chapter is thus to present a new model of signal that natively integrates both rythmic and harmonic information.

Chapter 18

Harmony & Rhythm Approach

18.1 Introduction

In this chapter we present a new model of signal, which natively includes both rhythmic and harmonic information. One of the fundamental idea of our model is that rhythm can be included in the representation thanks to an adaptive sampling. More precisely, the signal is sampled at dates that are musically meaningful. These can typically be determined by the salient peaks of an onset function. Each of these dates is then associated with a piece of information that locally characterize the signal. This can include a wide variety of local features with the possibility to concatenate several of them. In our work, we have focused on the use of a local harmonic information extracted from the chromagram. We have designed a search strategy according to this novel model of signal. Similarly to the one presented in the context of chords transcription, the search consists of a two-steps process. The first one relies on an indexing scheme that allows to efficiently target the best candidates for a match. The second step is a finer comparison performed on the reduced set of candidates.

The chapter is organised as follows. In a first section, we present in detail the model of signal that we have invented. This includes the definition of what we call the states of the signal, followed by the description of the onset function that allows to determine the times of occurrences of the states and finally the description of the harmonic local information that is included in the states. In a second section, we define the distance between two signals modelled according to our methodology. Our use-case indeed requires to have a measure of the musical similarity between two signals. As for the previous system, the distance we suggest is based on dynamic programming. In a third part, we present our search strategy. The latter is designed to retrieve in an efficient way a set of candidates from the database which includes the closest reference to the unknown signal. Finally, we present some experimental results.

18.2 The Signal Model

18.2.1 Model in states

The model that we propose relies on the extraction of musically significant instants in the music. In the following, we call these instants $t_1, t_2, ..., t_n$ the *dates of interest*. Our method then consists of associating to each interest date t_k an additional piece of information i_k. The nature of the information can be diverse and varied. It can include multiple traditional features that are concatenated. As a result, the mathematical nature of i_k is not restricted: it can be a vector, a matrix, a function... There are two main requirements concerning the attached information i_k. First, it must characterise the signal locally around the date of interest t_k. Second, it must include sufficient information in order to permit the computation of the indexing keys such as described in section 18.4.

Let us note here that the diverse transforms that are used to compute the dates of interests and the local information may involve different windowing of the signal. Typically, one does not use the same kind of windows for the extraction of harmonic information and for the extraction of rhythmic information. As a result, a particular care must be taken with the synchronising of the different transforms. Notably, the implementation has to avoid the rounding errors in the calculation of the time indexes: these can lead to a temporal drift between the transforms. It is also preferable that all transforms use centred windows so that the information processed belongs to the same region of signal.

We define the *state* s_k as the association of the date of interest t_k and its attached local information i_k.

$$s_k = (t_k, i_k)$$

The representation of a signal is finally given by the sequence of its extracted states $\{s_1, s_2, ..., s_n\}$.

18.2.2 Onsets

In our method, the *dates of interest* are localised thanks to an onset detection function. Here again, the onset detection function is merely an input to our identification strategy. The issue, however, receives a specific and active attention from the community, as illustrated by the presence of a specific contest "Audio Onset Detection contest" in the MIREX evaluation. Our contribution does not consist of extensively describing and studying the field. We give a short description of the principles used in most state of the art methods and then detail the specific implementation we made.

Klapuri [Kla99] gives an interesting definition of the onsets: *"We use the term onset detection to refer to the detection of the beginnings of discrete events in acoustic signals. A*

percept of an onset is caused by a noticeable change in the intensity, pitch or timbre of the sound [...]. A fundamental problem in the design of an onset detection system is distinguishing genuine onsets from gradual changes and modulations that take place during the ringing of a sound. This is also the reason why robust one-by-one detection of onsets has proved to be very hard to attain without significantly limiting the set of application signals." Let us additionally note that the above definition includes the detection of the start of harmonic notes (such as the start of a violin note) as well as the percussive hits from a drumkit. In the first case, only precise frequencies of the spectrum are impacted (namely the fundamental frequency of the note and its harmonics) whereas the second case generally involves an activity in virtually all frequencies.

Recent works in the domain all tend to use the same basis: projection of the signal on separate frequency bands, detection of the spectral variations in each band, integration of the results obtained in each band. The band-wise processing allows to deal with the fact that some of the onsets have an impact that is restricted to a particular frequency zone. Working on smaller frequency zones rather than tackling the entire spectrum at once makes the detection of these localised changes easier. The projection of the signal on the frequency bands is achieved thanks to filterbanks ([Kla99, Sch98]), the computation of the Short-Time Fourier Transform spectrogram ([Lar03, ARD07]) or the reassigned Short Time Fourier Transform spectrogram ([Pee07]).

These guidelines led us to the following implementation of an onset detection function. In a first step, the magnitude of the spectrogram of the signal is computed.

$$S_{(k,t)}^{FFT} = \left| \sum_{n=1}^{N} w(n-t)x(n)e^{-2j\pi\frac{k}{N}n} \right|$$

A pre-processing of the spectrogram mimicking the auditory nerve response is commonly suggested in the papers of the state of the art. It consists of a temporal filtering and a log compression. The filtering step masks the rapid variations in the spectrogram while preserving the sharp attacks. Taking the log of the spectrogram allows to remap the amplitudes of the signal on a scale that is closer to our perception. In a log-scale as well as in the human auditory system, the strength of an attack is perceived relatively to the surrounding amplitude level: a given shift in amplitude will have less impact if the starting point is a high amplitude level than if it is a quiet level. On each frequency band of the spectrogram

we apply the temporal filter whose transfer function is given by [ARD07]:

$$\Phi(z) = \frac{\alpha + \beta - (\alpha z_2 + \beta z_1 z^{-1})}{1 - (z_1 + z_2)z^{-1} + z_1 z_2 z^{-2}} \quad \text{with} \quad \begin{cases} T_1 = 15\text{ms} \\ T_2 = 75\text{ms} \\ \alpha = 1 \\ \beta = 5 \\ z_1 = e^{-1/T1} \\ z_2 = e^{-1/T2} \end{cases}$$

Then we log-compress the smoothed spectrogram $\widetilde{S}^{FFT}_{(k,t)}$:

$$G(k,t) = \log(\widetilde{S}^{FFT}_{(k,t)})$$

The Spectral Energy Flux is obtained by estimating the temporal derivative of the pre-processed spectrogram. Since the differentiation process is at the heart of the strategy of the onset detection, it is suggested in [ARD07] to be particularly careful with the calculation of the temporal derivative. Estimating the derivative of a discrete function has nothing of a trivial problem. The loss of information that results from the sampling process indeed makes the tangent estimation a tough issue. One can of course opt for a first order approximation (called the finite difference approximation) that simply consists of calculating $S(.,t+1) - S(.,t)$. This, however, does only give a rough approximation of the derivative. The use of higher order filters provides a much closer approximation to an ideal differentiation. The underlying idea of these methods is to find a polynomial that best fits the discrete function. The derivative is then estimated on the basis of this polynomial. The method that is retained is the use of a differentiator filter of order $2L$ based on the formula for central differentiation by Dvornikov [Dvo07]. The filter is given by the following transfer function:

$$\Psi(z) = -g(L) - g(L-1)z^{-1} - ... - g(1)z^{L-1} + g(1)z^{L+1} + ... + g(L)z^{2L}$$

with $g(k)$ defined by:

$$g(k) = \left(k \prod_{\substack{j=1 \\ j \neq k}}^{L} \left(1 - \frac{k^2}{j^2}\right) \right)^{-1}$$

The application of this filter on each frequency band of $G(k,t)$ gives an estimate of its differential with respect to time, which corresponds to the Spectral Energy Flux.

$$E(k,t) \approx \frac{\partial}{\partial t} G(k,t)$$

Since only the onsets are looked for, the next step consists of removing the impact of the offsets (disappearance of an event leading to negative values in the differential). This can be done thanks to a half-wave rectification.

$$E^+(k,t) = \begin{cases} E(k,t) \text{ if } E(k,t) \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Finally the information is integrated across the bands in the following way:

$$o(t) = \sum_k (E^+(k,t))$$

thus leading to a global onset detection function that shows high values at instants with a meaningful amount of change in the spectrogram in one or several frequency bands.

Our model needs the locations of discrete instants in the signal. This can be done thanks to the application of a peak-picking technique on the onset detection function. In order to ease this process, we apply beforehand a peak-widening filter that consists of convolving $o(t)$ with a half cosine, defined for $t \in [0;\tau]$ by:

$$h(t) = \frac{1}{2}\left(1 - \cos\left(\pi \frac{t+\tau}{\tau}\right)\right)$$

with $\tau = 25$ms. A classical peak-picking technique that looks for local maxima while respecting a density criterion is finally performed on the filtered onset function. The time locations of the extracted peaks are defined as the *dates of interest*.

18.2.3 Local information

In our method we have chosen, in the continuity of the previously described identification method (see section 17), to focus on local information of harmonic type. It indeed seems to us that rhythm and harmony are quite complementary in the task of representing music. Besides, the state of the art provides us with efficient tools for the estimation of harmony.

The harmonic content that we extract as local information relies on the chromagram, computed in the fashion of section 17.3. The further idea is that we expect the harmonic content to be relatively stable between two successive *dates of interest*. Indeed if the onset detection is reliable, any change in the harmony should be detected as a peak in $o(t)$, which would generate a new date of interest. For this reason we propose the introduction of the feature $c_{k,l}$: the mean chroma vector at the left of time t_k. In practice, it is computed by temporally averaging the chroma vectors between t_{k-1} and t_k. In a similar fashion, we

introduce $c_{k,r}$ the mean chroma vector at the right of t_k. Finally, we consider as local information the concatenation of these two features:

$$i_k = (c_{k,l}, c_{k,r})$$

18.2.4 Illustration

Figure 18.1 shows the superposition of the graphical representations of an onset detection function (in white) and a chroma representation of the same signal. In our model, the dates of interest $\{t_1, ..., t_n\}$ would be given by the dates of the yellow triangles on the onset function (obtained by peak-picking). For each date, the left mean chroma vector (mean chroma vector between the previous triangle and the current) and the right mean chroma vector (mean chroma vector between the current triangle and the next) would be computed, associated to the date and the whole would be stored as one state. The advantage of this model is that it is compact but still contains both rhythmical and harmonic information.

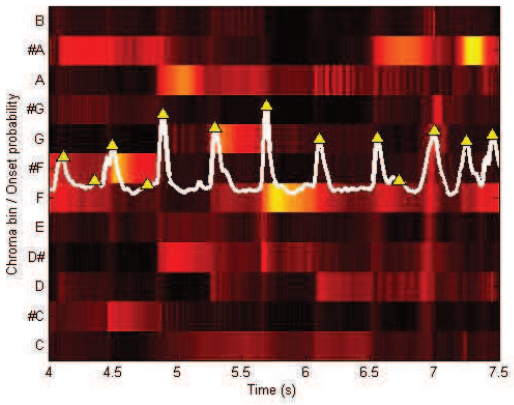


Figure 18.1: Illustration of the proposed model in states. The white curve figures the onset detection function. The local maxima extracted through peak-picking are made explicit by the yellow triangles. The background is the chromagram of the same signal with the higher energies figured by the lighter colours.

18.3 Distance

18.3.1 Introduction

Now that the model has been made explicit, there is a need to define the distance between two signals i.e. the distance between two sequences of states. For the same kind of reasons

as the ones detailed in section 17.4 we suggest the use of a dynamic alignment. Our task is then to align two sequences of time-localised states. For that matter, we can note that Gillet proposes an alignment strategy that fits the issue in [GR05]. His model is however restricted to the representation of drum loops and does therefore not cover the harmonic aspect of the matching. We consequently propose a meaningful extension of his work to the general case of music signals. We firstly present the motivations for the choice of a dynamic-programming based distance, then we present the detailled computation of the distance between two sequences of states.

18.3.2 Motivations for a dynamic-alignment based distance

The reasons that account for the choice of an approximate matching scheme, such as a dynamic alignment, are the following. A state corresponds to an onset location together with the surrounding harmonic content. Examining both cases of approximate and exact matching leads to the conclusion that the representation will certainly be distorted.

In the context of exact matching, the features that we use probably lack robustness to the usual post-processing distortions. We can indeed imagine that the harmonic representation (mean chroma vectors extracted from the chromagram) will have its levels of energy moved depending on the equalisation and enhancers used. As for the onsets, the presence of limiters and amplitude compressors in the processing chain may be very harmful for the quality of the onset detection. The latter will of course also be very sensitive to the additive impulse noises.

In the case of approximate matching, it is all the more clear that the harmonic features will vary. We hope for a similar harmonic colour between one performance and the other but the change of instruments, playing conditions, reverberation, arrangement... are very likely to result in different values for the harmonic feature. As far as the onsets are concerned, we must be aware that some artists will sometimes modify the rhythm of one title when playing live. Typically, they will play a chord repeatedly at the end of a phrase whereas the studio version contains a steady chord. Conversely, when playing a title at the guitar the artist may simplify some rhythms that are quite hectic in the studio version. As a result, from one version of a music title to the other, we see that some of the states may have disappeared (removal or addition of an onset) and that the preserved ones may have their harmonic features modified, while hopefully keeping the same colour.

18.3.3 Dynamic alignment

Let the first sequence of states be denoted by $\{s_1...s_m\}$ and the second one by $\{\sigma_1...\sigma_n\}$. When dynamically aligning two sequences of states, one has to define three types of penalties.

- $f^d(s_i)$: penalty assigned to the deletion of state s_i
- $f^i(s_i)$: penalty assigned to the insertion of state s_i
- $f^s(s_i, \sigma_j)$: penalty assigned to the substitution of state s_i by state σ_j.

In our implementation f^d and f^i are both taken constant and equal to 0.3. Knowing that $s_i = (t_i, (c_{i,l}, c_{i,r}))$ and $\sigma_j = (\tau_j, (\gamma_{j,l}, \gamma_{j,r}))$, we define f^s by:

$$f^s(s_i, \sigma_j) = \cos(c_{i,l}, \gamma_{j,l}).\cos(c_{i,r}, \gamma_{j,r}).\mathrm{e}^{\frac{|t_i - \tau_j|}{c}}$$

The first cosine term penalises the resemblance of the mean chroma vector at the left of s_i with the one at the left of σ_j. The second cosine term does the same for the mean chroma vector at the right. The exponential term penalises the timing error between the occurrence of state s_i and the occurrence of state σ_j.

Dynamic programming then consists of iteratively filling a scoring matrix D in the same way as in section 17.4. The score of the alignment of $\{s_1...s_m\}$ with $\{\sigma_1...\sigma_n\}$ is finally given by $D(n, m)$.

18.4 Keys and Search Strategy

18.4.1 Introduction

For the same reason as in the chords transcription system (see section 17.5), it is not conceivable to test one unknown signal against the whole reference database thanks to the distance methodology described above. We thus need to set up an indexing strategy that will speed up the search and make the algorithm scalable.

The search method we propose consists of extracting index keys from the representation in states that we introduced. The keys from the unknown signal are used to query an index that contains all the keys extracted from the references. By aggregating the index outputs obtained with the different keys of the unknown excerpt, we are able to determine the references that most likely correspond to the unknown signal. As we will see, the proposed keys are not very discriminative. This is a necessary condition for the ability of the index-based search to find the correct reference even when the unknown signal is the same title as the reference but

in a different version. The role of the index-based search is therefore limited to outputting a set of candidates that possibly match the unknown signal. These candidates are finally compared to the unknown signal thanks to the dynamic-alignment based distance described above.

18.4.2 Framing strategy

The broadcast has been previously transcribed as a long sequence of states s_i. The first step of the search consists of extracting from the entire sequence the frames that will be searched for. The framing is chosen according to the same considerations as in section 17.5.2. The subtlety here is that, given the non-uniform sampling of the model, a given temporal length does not correspond to a constant span of states indexes. The frames must consequently be designed with respect to the *dates of interest* assigned to the states.

Typically, in our implementation we chose a frame length of $L_a = 120s$. In order to ensure the whole capture within a frame of a 60s reference, the hop is taken equal to $t_{hop} = 60s$. This means that a frame starting at state s_i will end at state s_l such that:

$$s_l = \sup_{j \in \mathbf{N}} \{ s_j \,/\, t_j \leq t_l + L_a \}$$

Similarly, the following frame will start at state s'_i such that:

$$s'_i = \sup_{j \in \mathbf{N}} \{ s_j \,/\, t_j \leq t_l + t_{hop} \}$$

For the sake of mathematical rigour, let us additionally clarify that the order relationship on the set of states is naturally defined by: $s_i \leq s_j \Leftrightarrow t_i \leq t_j$.

18.4.3 Index keys

In this context we must define the indexing keys. Let us recall that the indexing scheme only processes exact queries. The crucial point is thus to set up distortion-invariant keys, while still sufficiently discriminating. The invariance to the distortions should handle the classical post-processing distortions (equalisation, pitch-shifting, ...) but also the recording of the same music title in different conditions (matching of similar items).

The idea is to index the references by their harmonic content. However, the local information that we defined, as such, is far too variable to provide a robust key. The traditional reasoning in this typical indexing issue leads us to the set up of a vector quantisation method. In the preceding identification system (see section 17), we did propose a projection of the

chroma vectors on a chords dictionary. This, of course, was a way of quantising the harmonic information. Here, however, we suggest the use of a much simpler quantiser.

We propose the following keys generation mechanism. For each state s_k one key is generated. The key is a binary version of the mean chroma vector at the right of t_k. Practically, for any bin b of the chroma vector, the binary chroma vector $\widetilde{c}_{k,r}$ is given by:

$$\widetilde{c}_{k,r}(b) = \begin{cases} 1 & \text{if } c_{k,r}(b) > \overline{c_{k,r}} \\ 0 & \text{otherwise} \end{cases}$$

with $\overline{c_{k,r}}$ the mean value of vector $c_{k,r}$.

Let us note that the possible lack of robustness of the quantisation process could be counterbalanced by the generation of a neighbourhood, similarly to what is done in BLAST (see section 17.5). Instead of calculating one single quantised vector $\widetilde{c}_{k,r}$, one would then generate a set of possible vectors by moving the bins that are close to the threshold. For now, we have not implemented such an extension since our experiments did not show that it was necessary.

18.4.4 Aggregation of the index outputs

The index is queried with keys that are quantised versions of the harmonic features. All the references possessing a state with a similarly quantised harmony is then output by the index engine. The output contains the identifier of the concerned reference as well as the date of interest of the state(s) containing this harmony. As for the previous methods, there is a need to aggregate these index outputs. This is particularly true in this case since the queries are quite unselective. One can actually expect that a meaningful number of references do possess a given harmony. Besides, this harmony has good chances to repeat several times within one reference containing it.

Similarly to the methods described in section 17.5 and 11.2, for one reference hit at least by one key we gather all the index outputs that involve this reference. More precisely, let k be the key extracted from state $s_q(k)$ whose date of interest is $t_q(k)$ in the query. Let us say that the same key k was extracted from several states of reference r with dates of interests $t_r^1(k), ..., t_r^M(k)$ during the learning stage. These pieces of information were consequently stored in the index. When querying the index with the key k, we thus get in output the identifier of reference r together with the dates $\{t_r^j(k)\}$. We gather these pieces of information by storing the couples $(t_r^1(k), t_q(k))$, ..., $(t_r^i(k), t_q(k))$ in a scatter plot dedicated to reference r (we build one scatter plot per reference output at least once by the index). We do the same for all the references that were output by the index when querying it with k and we iterate the process for all the keys extracted from the query.

The histogram process described in 11.2 can in fact be seen as the search for a linear relationship in the scatter plot (i.e. a linear relationship between the times of occurrences of the keys in the unknown signal and their times of occurrences in the reference signal). However, this search is limited to straight lines having a unitary slope. In our context, we do not expect the actual keys to have a unitary linear relationship between their times of occurrences. This firstly comes from the fact that different versions of a same title may be performed with slightly different tempos. Besides, since in the approximate matching context we perform the identification process on wide frames, the time-stretching effect sometimes used in post-processing also induces a perceptible effect on the linear relationship.

Let us indeed consider that the query is an excerpt of the reference r starting at time t_0. Let us also consider that the query is time-stretched by a factor κ (either because of some specific post-processing or because the query is another record of the same song with a slightly different tempo). Then, the key k extracted from the query at time $t_q(k)$ should be retrieved in the reference r at time $\kappa(t_q(k) - t_0)$. This means that, in the scatter plot of reference r, all these corresponding keys produce dots that are located on the straight line:

$$Y = \frac{1}{\kappa}X + t_0$$

Though, the scatter plot will also contain a meaningful number of dots that are outside this line. These correspond to keys that are found in the query and that occur several times in the reference. We must indeed keep in mind that the keys we defined are not very discriminative. Figure 18.2 shows an example of such a scatter plot.

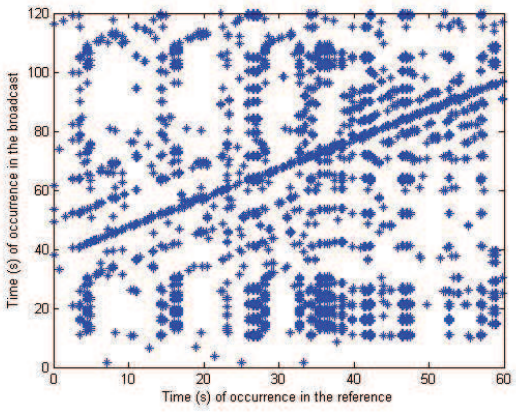


Figure 18.2: Scatter plot aggregating the index outputs for one reference. Each dot with coordinates (x, y) corresponds to a key that was extracted at time $t = x$ in the reference and that occured at time $t = y$ in the broadcast frame.

In order to locate the most likely linear relationship in the scatter plot, we use Hough's algorithm [DH72]. This accumulation technique allows to find the straight line that contains the highest number of points in the graph. The nice thing with this technique is that the slope and the intercept are jointly estimated in a very efficient way. This is of particular interest since scalability is still at the heart of our problem.

The implementation of Hough's algorithm relies on the setup of an accumulator. The latter is a two-dimensional array that we name $\mathcal{A}$. The first dimension is representative of the slope parameter. The second dimension stands for the intercept. In $\mathcal{A}(m, p)$, one reads the number of points of the graph that belong to the straight line $Y = mX + P$. If the K dots of the scatter plot are denoted by $d_k = (x_k, y_k)$, the processing is the following:

for m *from* m_{min} *to* m_{max} **do**
  for k *from 1 to* K **do**
    $p = y - mx$;
    $\mathcal{A}(m, p) \leftarrow \mathcal{A}(m, p) + 1$;
  end
end

In order to locate the most likely straight line in the graph, we only have to look for the maximum value in the accumulator. Its argument (m_0, p_0) gives the parameters of the line.

In the end, we have, for each reference, the parameters of the best line (corresponding to the time-stretching ratio κ and the start time t_0) and the number of dots (i.e. the number of keys) that match this line. The M references with the highest number of matching keys are considered as the M best matches to the query. They are selected as candidates.

18.4.5 Fine matching of the candidates

Similarly to the method proposed in section 17.5, the indexed search is followed by a fine matching step which consists of calculating the distance between the candidates and the query. It is here quite obvious that the processing involved in the distance calculation allows a much more refined evaluation of similarity than the study of the correlation of the quantised harmonic keys. Let us note for that matter that the model can easily be extended through the addition of features as local information in the states. These would not be taken into account in the key generation process and in the indexed search, but they could easily be integrated in the distance calculation. One final recall is that the dynamic programming involved in this step is CPU costly but, once again, we are working with a reduced set of candidates.

18.5 Experiments and Results

We conduct the exact same experiment as for the chords transcription method (see section 17.6). The analysed stream corresponds to 24 hours of the French radio 'RTL' and contains 8 broadcasts of references that lay in the field of approximate matching. The database is composed of 2400 excerpts of 60s of music titles.

The results are presented under the same form of a ROC curve (see Figure 18.3), with no other post-processing than a varying threshold that parametrises the trajectory of the curve. For comparison, the graph still shows the ROC curve of an exact matching algorithm from the state of the art. We also display the best results we had with the chords sequence identification system.

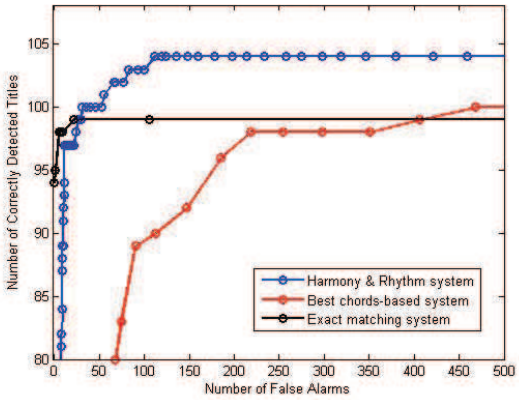


Figure 18.3: Results of the Harmony & Rhythm identification system under the form of a ROC curve (in blue). The X coordinate gives the number of false alarms whereas the Y coordinate gives the number of correctly detected titles. The graph shows the ROC curve of a traditional system for exact matching in black and the one of a chords-based system in red.

The results show that the method, similarly to the one relying on chords transcription, is still able to go beyond the traditional limitation of exact matching. However, we can see that it succeeds in this task with a much lower number of false alarms than the preceding one. This tends to show that by adding a further rhythmic component in the model, we could make the fingerprint a lot more discriminative while still being very robust. The results achieved indeed show that the method is robust to the traditional post-processing distortions but it can also match different versions of one same title. For that matter it is interesting to note that, in the algorithm's present state, the scores do not classify the broadcasts according to their level of perceived distortion. Indeed, some items that are sounding really different from the reference (different version with different instrumentation) have a better score than

others that are only post-processed (strong amplitude compression but the version is the same). This behaviour could easily be modified by extending the features list (for example by adding timbra descriptors) that is taken into account in the local information of the states and in the final distance computation.

Chapter 19

Conclusion

In the part "Exact Matching" we did set the focus on a method that turned out to be an exact matching algorithm. The objective of the current part was to propose new methods that have the ability to handle the problem of approximate matching. Given this objective, we took a particular care in the choice of the features. These indeed have to be representative of our perception of music.

The first method we proposed is based on an automatic transcription of music in chords sequences. This kind of technology is nowadays well established and achieves sufficiently high scores in order to be used as a front-end in our audio-identification paradigm. We however always kept in mind that the chords transcription is error-prone and we adapted our research mechanisms accordingly. More precisely, our search strategy is inspired from the field of approximate string matching. The resulting system was tested with the same Quaero evaluation protocol as for our initial exact matching system but we took care of including in the corpus some broadcasts which were only similar to their corresponding references. These broadcasts were actually live performances of music titles whose studio version was in the database. The good news is that the results clearly show that the method has the ability to handle the approximate matching case. It is clear on the displayed ROC curves that the exact matching method is confined to a domain whose borders are overstepped by the chords-sequence method. The other side of the coin however is that this method generates much more false alarms. One nice characteristic of the initial system is indeed that it produces very few false alarms. The precise study of these false alarms however reveals that they rely on confusions that are harmonically justified. The mistaken output generally corresponds to a chords sequence that is very close to the input sequence. Although this cannot be considered as a correct identification in the field of audio-identification, it can still present some interest for other applications.

In order to improve the results, we tried to develop another model with additional in-

formation in the representation. The challenge of course consists of making the model more precise while keeping its ability to handle the similar items. The chosen solution consisted of adding a rhythm aspect in the modelling. In order to integrate nicely both pieces of information that are rhythm and harmony, we proposed a model based on an adaptive sampling. This induced the necessity to develop an appropriate search strategy. We tested this approach on the same evaluation framework as the chords-sequence based approach. The results are quite satisfying since the approach allowed to dramatically reduce the number of false alarms while still having the ability to match similar items. A deeper analysis of the results showed that, surprisingly, the detections with the lowest score are not necessarily the least similar. As far as we could see, it seemed that the most penalised items are the one which underwent an important dynamic compression. In this case the rhythmic information is quite badly estimated. In other words, the weakest link of the algorithm in its present state is the onset detection function. For that matter it would be worth working on onset functions with an increase robustness to compression.

Part VI

General Conclusion

Chapter 20

Synthesis

Throughout this work, we have explored various audio-fingerprint models associated with indexed-based search strategies for the purpose of large scale audio-identification.

We have started with a description of the audio-identification use-case, which consists of automatically retrieving the meta-data associated to an unknown sound. We have then made precise that audio-fingerprint algorithms meet this objective by extracting from the audio signals a characteristic fingerprint. By learning beforehand all the fingerprints of a set of references, the algorithm is then able to identify any signal that belongs to this set. This is done by extracting the fingerprint from the unknown signal and looking for the closest learnt fingerprint. We have completed our description of the use-case by emphasising the main stakes when designing an audio-fingerprint algorithm. First, the algorithm must be able to identify any signal that corresponds to a reference, even if it has undergone a series of distortions. Second, the algorithm must be able to manage a very large set of references (typical industrial databases include hundreds of thousands of music titles). As far as the distortions are concerned, we have made a specific distinction between post-processing distortions, which are the ones that occur in the transmission channel of a given music recording (dynamic compression, equalisation, pitch-shifting, additional noise...) and the variations that occur when we study two different recordings of one same music title. In the first case, we talk about *exact matching* whereas we use the terminology *approximate matching* in the second case.

In the second part, we have specified the use-case of interest in this work: the automatic annotation of broadcast streams. In this context, the input to the algorithm is a continuous stream that contains, amongst others, the broadcast of some references which were learnt beforehand by the algorithm. The use of real-world radio streams in our experiments ensures a meaningful level of post-processing distortion in the corpus. The radio stations indeed apply numerous processings to the sound before broadcasting. The task of the algorithm is

to detect all the broadcast references. We have finally proposed an associated scoring metric. The set up of this evaluation framework, associated with the definition of the corpus and of the metrics, is the object of the following paper.

- Mathieu Ramona, Sébastien Fenet, Raphael Blouet, Hervé Bredin, Thomas Fillon and Geoffroy Peeters, "A Public Audio Identification Evaluation Framework for Broadcast Monitoring", *Applied Artificial Intelligence: An International Journal*, vol. 26, no. 1-2, pp. 119-136, February 2012.

In the next part, we have proposed an exhaustive study of the state of the art in audio-fingerprint. We have identified four groups of methods, according to the way the fingerprint is calculated. In the first group, the fingerprint is directly extracted from the temporal representation of the signal. In the second group, the methods extract short-term features from the spectrogram of the signal. The underlying idea is that these short features will not be distorted. The methods consequently use these features as keys in an exact index scheme. The methods from the third group rely on the extraction of long term characteristics of the spectrogram. These are subject to variations when the signal is distorted so that the methods are associated with approximate search strategies. In the last group, the spectrogram is transcribed in a string of symbols thanks to a vector quantisation scheme. The search for a reference similar to the unknown signal subsequently becomes a string matching problem.

Amongst the methods from the state of the art, we have focused on the one proposed by Wang in [Wan03], commonly referred to as "Shazam's method". We have reminded the reader of the principles of the method: binarisation of the spectrogram thanks to the extraction of local maxima then constitution of index keys by grouping the maxima in pairs. We have additionaly proposed a post-processing step that is not described in the original paper and which allows to drastically reduce the number of false alarms. The part ends with an exhaustive testing of our implementation of the method. In these experiments, the method actually shows a lack of robustness to the pitch-shifting distortion.

Our third part is dedicated to the proposition of improvements over our initial implementation of [Wan03]. These include two separate aspects: the proposition of different signal models for the fingerprint and the adaptation of the method to an extended functional perimeter. At first, we have elaborated a tracking method that processes the outputs of our automatic annotation algorithm. The resulting system has the ability to track a reference throughout its broadcast. We have then proposed the use of the Constant Q Transform for the computation of the spectrogram. This, in turn, allows the definition of a new fingerprint that shows an increased robustness to pitch-shifting. We have also proposed an optional pruning step that speeds up the search. These contributions are described in the two following papers.

- Sébastien Fenet, Yves Grenier, and Gaël Richard, "Une empreinte audio à base de CQT appliquée à la surveillance de flux radiophoniques", in *Proceedings of the Groupe d'Etudes du Traitement du Signal et des Images (GRETSI)*, Bordeaux, France, September 2011.

- Sébastien Fenet, Gaël Richard, and Yves Grenier, "A Scalable Audio Fingerprint Method with Robustness to Pitch-Shifting", in *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, Miami, USA, October 2011, pp. 121-126.

Furthermore, we have proposed an adaptation of the architecture of our modified method to deal with a different use-case: the automatic detection of recurrent motives within a stream. We have also taken advantage of this work to try the approach with a radically different model of fingerprint, which is based on a sparse decomposition of the signal in a redundant dictionary. This work constitutes the object of the following paper.

- Sébastien Fenet, Manuel Moussallam, Yves Grenier, Gaël Richard, and Laurent Daudet, "A Framework for Fingerprint-Based Detection of Repeating Objects in Multimedia Streams", in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania August 2012, pp. 1464-1468.

In the following part, we have focused on a quite different issue. We start from the observation that, to our knowledge, virtually no method from the state of the art in audio-fingerprinting deals with the approximate matching use-case. We however believe that there is a real research interest as well as practical applications for this issue. We have thus dedicated the last part of our work to the proposition of methods that meet the objectives of approximate matching.

At first, we propose an audio-fingerprint algorithm which is based on the transcription of the signal in chords sequences. The chords transcription can equivalently be seen as a vector quantiser. It thus transcribes any audio signal into a string. Amongst the available techniques from the domain of approximate string matching, we have chosen to adapt the BLAST strategy [AGM$^+$90] to our context. In the end, we have an audio-fingerprint algorithm that uses sequences of chords as fingerprints, and whose fast search strategy relies on an approximate string matching method. A paper reporting this work will be written.

- *In Preparation:* Sébastien Fenet, Gaël Richard, Yves Grenier, "A Chords-Transcription Based Audio-Fingerprint Method For Approximate Matching".

The final approach relies on an innovative representation of the signal. The latter is composed of compact states that include both rhythmic and harmonic information. Based

on this representation, we present an efficient search strategy that allows to quickly find the references most likely corresponding to an unknown signal. The representation of the signal and the search strategy have been designed in a way that makes the approach robust to the change of version of a music title. A pending patent describes the method and a paper on the topic is in preparation.

- Sébastien Fenet, Yves Grenier, and Gaël Richard, "Génération d'une Signature d'un Signal Audio Musical", FR Patent Pending 1351752.

- *In Preparation:* Sébastien Fenet, Yves Grenier, Gaël Richard, "An Extended Audio-Fingerprint Method with Capabilities for Similar Music Detection".

Chapter 21

Conclusion and Perspectives

21.1 Exact Matching

As far as exact matching is concerned, the following observations can be made. Our original implementation of [Wan03] shows a weakness in the specific case of pitch-shifting. This has been corrected thanks to the modification of the fingerprint model. Once this has been done, one can observe that the results obtained in the real-world experiments are barely improvable. Besides, it is also observable in our report that the performance of IRCAM's algorithm is of the same order. In spite of the lack of wide scale evaluation campaign in the domain, one can infer that most methods at the level of the state of the art somehow have the ability to reach such scores.

In conclusion, I would say that nowadays challenge does not lie anymore in the improvement of the scores on such a use-case. While mentioning a direction which does not represent a challenge anymore for the researchers of the domain, let us mention two other issues that should not be considered as research challenges.

First, we should say that reducing the size of the query does not constitute a challenge. The same goes for increasing the compactness of the fingerprints. We have indeed seen that the state of the art offers a wide variety of methods, some of them use very localised features (as in [HKO01]) while others use long term descriptors (as in [RP11]). Both above mentioned problems can thus be solved simply by choosing the most adapted strategy within the state of the art. Second, one can also think of the scalability challenge, which involves the set of questions: can we prune the database? can we perform parallel computing with the approach? can we divide the search with a front-end database (containing the most likely queries) and a back-end database? These issues, however, lie at the border between research and engineering. In other words, once a "sufficiently scalable" approach has been designed, the tuning of the code, the adaptation of the parameters and a good knowledge of computers

should allow to move to industrial databases.

On the other hand, a problem that is not explicitly studied in the state of the art is the detection of background music. It would be interesting to focus on signals with a very low Signal to Noise Ratio, such as: a phone conversation which takes place in a pub with a background music, the background soundtrack of a TV show when the actors speak, the broadcast of a title as background sound while the presenter speaks in a radio show, ... The fingerprint extraction should then take into account the fact that most of the spectral information is not linked with the music to identify. Two ways could be considered to this end. The first solution would consist of using source separation techniques in order to isolate the target information. Then a classical audio-fingerprint model would be applied to the isolated music signal. The second, more integrated, solution would consist of designing a fingerprint model that inherently isolates the most interesting bits of information. This could be done according to statistics consideration (for example: if the music is a background signal, its spectral components should show a lower magnitude).

Another way of extending the problem of approximate matching lies in our second use-case (automatic detection of recurrent motives). Our current system has the ability to detect the redundancies at a frame level. A rather naive clustering strategy allows to group the frames in what should correspond to songs. Our experiments have however showed that the clustering was too simple for the discrimination of songs versus other long repeating sections of streams. This essentially occurs because the clustering strategy we proposed treats each cluster independently. Designing a clustering technique that takes into account, at once, all the repeated sections that include the same set of frames should allow a more realistic discovery of the songs. More precisely, for each cluster of frames that have been repeated, the algorithm would study the structure of these repeated frames (thanks to a dynamic programming approach for instance) through all their different repetitions. In spite of refining the determination of whether a cluster corresponds to a song or not, this strategy should make the algorithm able to discover and store the various edits of one same title. Finally, having such a strategy available, the algorithm could automatically clean its database. The latter would then only contain the sections of broadcast that correspond to music titles. The overall memory of the system could in consequence dramatically increase, allowing the algorithm to process much more days of broadcast with the same database.

21.2 Approximate Matching

When tackling the problem of approximate matching, our aim was to provide new fingerprint models, associated with their search strategies, that would be able to identify items that are

similar to one of the references. Our first (chords-based) approach has proved to be able to do so. However, this has come at the cost of a dramatic increase in the number of false alarms. Let us indeed recall that the exact matching audio-fingerprint algorithms have a very low rate of false alarms.

In a sense, the result was predictable. The description of a music based on its succession of chords is a rough simplification of the musical content. On the one hand, this approximation permits the detection of similar item. On the other hand, it is not rare that two distinct music titles possess the same harmonic progression.

For that matter, it is interesting to note that the false alarms produced by the algorithm are harmonically coherent. Two signals that are detected as similar share a common chord progression. For this reason, it would be interesting to study the issue further. One could notably try to cluster the references according to their distance in the chords sequences space. When identifying an unknown signal, one would then not obtain a single reference that matches the signal but a cluster of references that possess a common harmonic configuration. Such a system could for instance be used as an automatic recommendation algorithm for the DJs. Given the distance model that we set up, it should indeed be possible to superimpose two titles from the same cluster in a nice sounding way.

Our second approximate matching approach has proved to possess the same ability in detecting the similar items as the first one, with the additional benefit of generating a much lower number of false alarms. Let us recall that this method relies on a novel representation of the signal that includes both rhythmic and harmonic information.

The conclusion is that the introduction of a rhythm component in the model could make the fingerprint much more discriminative while keeping its robustness to a change of version of a music title.

It is however to be noted that given the current search strategy and distance model, there are at least two variations against which the method will not be robust. First, the harmonic features we derive are not robust against transposition. There are numerous ways to render a method robust to transposition, some of which can be found in [SGHS08] or [HK03]. It would thus be interesting to see if we can adapt our model accordingly. The other issue that comes with cover versions is the change of structure. The search for straight lines in the scatter plots of our current search strategy ensures that the method cannot deal with different versions of one title in which the structure has been modified. For this matter, one would have two options. The first one would consist of using sufficiently short query frames so that the structure problem does not show. The second option would consist, in the scatter plot analysis, of looking for more complex patterns than a straight line. The issue, however, is to keep the scatter plot processing as efficient as possible, since it is a necessary condition

for the scalability of the approach.

While talking about scalability, it would be worth studying the impact of the distances that we introduced. Let us indeed recall that the approximate matching models we suggest are both associated with perceptually meaningful distance measures. The space in which the fingerprints live is then a metric space. This can in turn be used in the search step. Let us for instance imagine that we have pre-processed the references so that we know the distance between any two references. When searching an unknown signal, one could then compute the distance between this signal and one given reference. Imagining that this reference is found to be far from the query signal, one could eliminate from the search, thanks to the triangle inequality, all references that are close to this initial reference. Such a process would lead to a radically different search scheme. It would be interesting to study the scalability of such an approach.

In a more global point a view, we can note that the perspectives in the domain of approximate matching stay quite open. Indeed, our study of the state of the art has exhibited only a few works dealing with the detection of similar items within large reference databases. It is thus interesting to consider the proposition of new fingerprint models with such characteristics. These should ideally allow the algorithms to increase their precision in the detection problem (i.e. reduce the number of false alarms) while keeping the ability to detect similar items. An alternative to this compromise would consist of developing algorithms that are able to tell whether the detected item is an exact match or an approximate match. With such a system, one would expect a low number of false alarms for the exact matches whereas the tolerance would be wider when dealing with approximate matches.

Appendix A

Working with industrial data

This work has been carried out in the context of a partnership with a company. As such, it has appeared very beneficial to adopt some best practices. The latter can indeed allow to better deal with: the meaningful volume of data that has been used in this work, the regular transmission of new data, the adaptation to the changes of API of the partner, the adaptation to the decisions taken with the other partners concerning the evaluations (size of the database, the scope of the analysed broadcasts, output format...). The work methodology adopted should thus include the following concepts.

Md5 checking Every exchange of data should always include an integrity verification such as the calculation of an md5 code. This ensures that none of the partner is working with corrupted data, which can cause meaningful wastes of time.

Symbolic links In our context, the industrial partner has sent us meaningful volumes of data that correspond to the reference database. This database is not always used as a whole: some steps like the testing of the evaluation protocol involve the use of a subset of the database. Some evaluation campaigns have also included different tasks with different sizes of databases. In order to modulate the database, several approaches can be considered. It can for example be handled in the fingerprint algorithm. However, we rather recommend the setup of a specific folder for each test, that contains all the files of the tested database. This methodology makes it easier to keep track of the successive testings. Though, since the concerned files represent a substantial weight, it would be a waste of space and time to make a different copy of each reference file for each test database! This is why we suggest the use of symbolic links. In the end, the reference files are physically stored in one single folder. Each test database is represented by a specific folder that contains symbolic links to the physical folder. In this way, we keep track of the database used in each testing, with no

waste of space and copy time.

Network hard-disks Since the calculations involved in the fingerprint tasks can be really heavy, it must be considered that the evaluation campaigns may require the use of several computer units. In order to ease the setup of these, we recommend to store all the data linked with the testing on network hard-disks.

Separation of the APIs It is a general good practice to ensure a good modularity in any code. In our particular case, the fingerprint algorithm must get the data in the files provided by the industrial. This can be done thanks to the specification of an API together with the partners. However, experience has shown that these APIs may evolve along with the project. The consequential modifications in the code may be substantial and hard to debug if the issue has not been foreseen. This is why we recommend to explicitly separate all the bits of code that access the data provided by the partners. This modularisation limits the scope of the modifications when the data format evolves.

Bibliography

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool." *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, Oct. 1990.

[AHH+01] E. Allamanche, J. Herre, O. Hellmuth, B. Fröba, T. Kastner, and M. Cremer, "Content-based Identification of Audio Material Using MPEG-7 Low Level Description," in *International Symposium on Music Information Retrieval*, Bloomington, Indiana, USA, October 2001.

[ARD07] M. Alonso, G. Richard, and B. David, "Accurate tempo estimation based on harmonic + noise decomposition," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, pp. 161–161, Jan. 2007.

[BC06] S. Baluja and M. Covell, "Content fingerprinting using wavelets," in *European Conference on Visual Media Production*, London, UK, Nov. 2006, pp. 198–207.

[BCR07] M. Betser, P. Collen, and J.-B. Rault, "Audio Identification using Sinusoidal Modeling and Application to Jingle Detection," in *International Symposium on Music Information Retrieval*, Vienna, Austria, September 2007.

[BDP+05] C. Burges, D.Plastina, J. Platt, E. Renshaw, and H. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," in *International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, USA, Mar. 2005.

[Bel54] R. Bellman, "The Theory of Dynamic Programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[Ber99] S. M. Bernsee, "Time Stretching And Pitch Shifting of Audio Signals – An Overview," Aug. 1999. [Online]. Available: http://www.dspdimension.com/admin/time-pitch-overview/

[Bet08] M. Betser, "Modélisation sinusoïdale et applications à l'indexation sonore," Ph.D. dissertation, Telecom ParisTech, 2008.

[BME11] T. Bertin-Mahieux and D. P. W. Ellis, "Large-Scale Cover Song Recognition Using Hashed Chroma Landmarks," in *Workshop on the Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2011, pp. 117–120.

[BME12] T. Bertin-Mahieux and D. P. Ellis, "Large-Scale Cover Song Recognition Using the 2D Fourier Transform Magnitude," in *International Symposium on Music Information Retrieval*, Porto, Portugal, Oct. 2012.

[BMGC04] E. Batlle, J. Masip, E. Guaus, and P. Cano, "Scalability issues in an HMM-based audio fingerprinting," in *International Conference on Multimedia and Expo*, vol. 1, Taipei, Taiwan, 2004, pp. 735–738 Vol.1.

[BP92] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, Nov. 1992.

[BPJ03] C. J. C. Burges, J. C. Platt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 165–174, 2003.

[BPP05] J. Bello, J. Pickens, and S. Pauws, "A Robust Mid-level Representation for Harmonic Content in Music Signals," in *International Symposium on Music Information Retrieval*, London, UK, Sep. 2005.

[Bro91] J. C. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, January 1991.

[BYN96] R. A. Baeza-Yates and G. Navarro, "A faster algorithm for approximate string matching," in *Annual Symposium on Combinatorial Pattern Matching*, Laguna Beach, CA, USA, Jun. 1996, pp. 1–23.

[BYRN99] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[CB07] M. Covell and S. Baluja, "Known-audio detection using waveprint: Spectrogram fingerprinting by wavelet hashing," in *International Conference on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, USA, Apr. 2007.

[CBG+02] P. Cano, E. Batlle, E. Gomez, L. de C.T. Gomes, and M. Bonnet, "Audio Fingerprinting: Concepts and Applications," in *International Conference on Fuzzy Systems and Knowledge Discovery*, Singapore, November 2002.

[CBKH02] P. Cano, E. Battle, T. Kalker, and J. Haitsma, "A Review of Algorithms for Audio Fingerprinting," in *IEEE Workshop on Multimedia Signal Processing*, St. Thomas, Virgin Islands, USA, Dec. 2002, pp. 169 – 173.

[CBMN02] P. Cano, E. Battle, H. Mayer, and H. Neuschmied, "Robust Sound Modeling for Song Detection in Broadcast Audio," in *Audio Engineering Society Convention*, Munich, Germany, May 2002, p. 5531.

[CE10] C. Cotton and D. Ellis, "Audio Fingerprinting to Identify Multiple Videos of an Event," in *International Conference on Acoustics, Speech and Signal Processing*, Dallas, Texas, USA, 2010.

[Cob95] A. Cobbs, "Fast approximate matching using suffix trees," in *Annual Symposium on Combinatorial Pattern Matching*, vol. 937, Espoo, Finland, Jul. 1995, pp. 41–54.

[DH72] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[DR10] E. Dupraz and G. Richard, "Robust frequency-based audio fingerprinting," in *International Conference on Acoustics, Speech and Signal Processing*, Dallas,USA, March 2010, pp. 2091–2094.

[Dvo07] M. Dvornikov, "Formulae of numerical differentiation," *Journal of Concrete and Applicable Mathematics*, vol. 5, p. 77, 2007.

[EP07] D. P. W. Ellis and G. E. Poliner, "Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking," in *International Conference on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, USA, Apr. 2007.

[Fuj99] T. Fujishima, "Realtime chord recognition of musical sound: a system using Common Lisp Music," in *International Computer Music Conference*, Beijing, China, 1999, p. 464–467.

[GR05] O. Gillet and G. Richard, "Drum loops retrieval from spoken queries," *Journal of Intelligent Information Systems*, vol. 24, no. 2, pp. 159–177, 2005.

[HAH01] J. Herre, E. Allamanche, and O. Hellmuth, "Robust matching of audio signals using spectral flatness features," in *Workshop on the Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct. 2001.

[Her06] C. Herley, "Argos: automatically extracting repeating objects from multimedia streams," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 115–129, 2006.

[Hip13] R. Hipp, "SQLite Home Page," Mar. 2013. [Online]. Available: http://www.sqlite.org/

[HK03] J. Haitsma and T. Kalker, "Speed-change resistant audio fingerprinting using auto-correlation," in *International Conference on Acoustics, Speech and Signal Processing*, Apr. 2003.

[HKO01] J. Haitsma, T. Kalker, and J. Oostveen, "Robust audio hashing for content identification," in *International Workshop on Content-Based Multimedia Indexing*, Brescia, Italy, September 2001.

[HS05] C. A. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *Audio Engineering Society Convention*, Barcelona, Spain, 2005.

[HS08] A. Holzapfel and Y. Stylianou, "Rhythmic similarity of music based on dynamic periodicity warping," in *International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 2217–2220.

[JRLJ96] F. Johannesen, S. Raaschou, O. Larsen, and P. Jürgensen, "Using weighted minutiae for fingerprint identification," in *Advances in Structural and Syntactical Pattern Recognition*, vol. 1121, Leipzig, Germany, 1996, pp. 289–299.

[JU91] P. Jokinen and E. Ukkonen, "Two algorithms for approximate string matching in static texts," in *Annual Symposium on Mathematical Foundations of Computer Science*, vol. 520, no. 6, 1991, pp. 240–248.

[KCPD13] H. Khemiri, G. Chollet, and D. Petrovska-Delacrétaz, "Automatic detection of known advertisements in radio broadcast with data-driven alisp transcriptions," *Multimedia Tools and Applications*, vol. 62, no. 1, pp. 35–49, 2013.

[KHS05] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ser. CVPR '05, San Diego, CA, USA, 2005, pp. 597–604.

[Kla99] A. Klapuri, "Sound Onset Detection by Applying Psychoacoustic Knowledge," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 6, Washington, DC, USA, 1999, pp. 115–118.

[KM08] F. Kurth and M. Muller, "Efficient index-based audio matching," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 382–395, 2008.

[KP00] S.-R. Kim and K. Park, "A dynamic edit distance table," in *Annual Symposium on Combinatorial Pattern Matching*, Montreal, Canada, Jun. 2000, pp. 60–68.

[KPDC12] H. Khemiri, D. Petrovska-Delacrétaz, and G. Chollet, "Une empreinte audio à base d'ALISP appliquée à l'identification audio dans un flux radiophonique," in *Compression et Représentation des Signaux Audiovisuels*, Lille, FR, May 2012, pp. 139–144.

[Kur02] F. Kurth, "A ranking technique for fast audio identification," in *IEEE Workshop on Multimedia Signal Processing*, St. Thomas, Virgin Islands, USA, Dec. 2002, pp. 186–189.

[Lar00] J. Laroche, "Process for identifying audio content," Patent US6 453 252 B1, may, 2000. [Online]. Available: http://www.google.com/patents/US6453252

[Lar03] ——, "Efficient Tempo and Beat Tracking in Audio Recordings," *Journal of the Audio Engineering Society*, vol. 51, no. 4, pp. 226–233, 2003.

[LBG80] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[LcWC09] A. Li-chun Wang and D. Culbert, "Robust and invariant audio pattern matching," Patent US7 627 477 B2, December, 2009. [Online]. Available: http://www.freepatentsonline.com/7627477.html

[LcWSI11] A. Li-chun Wang and J. O. Smith III, "System and methods for recognizing sound and music signals in high noise and distortion," Patent US7 865 368 B2, January, 2011. [Online]. Available: http://www.freepatentsonline.com/7865368.html

[LCY+09] Y. Liu, K. Cho, H. S. Yun, J. W. Shin, and N. S. Kim, "DCT based multiple hashing technique for robust audio fingerprinting," in *International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, Apr. 2009, pp. 61 – 64.

[LMSS95] G. M. Landau, E. W. Myers, J. P. Schmidt, and P. Schmidt, "Incremental string comparison," *SIAM Journal on Computing*, vol. 27, pp. 557–582, 1995.

[Lou90] J. Lourens, "Detection and Logging Advertisements using its Sound," in *IEEE Symposium on Communications and Signal Processing*, Johannesburg, South Africa, Jun. 1990, pp. 209 – 212.

[LOX06] H. Lin, Z. Ou, and X. Xiao, "Generalized time-series active search with kullback-leibler distance for audio fingerprinting," *IEEE Signal Processing Letters*, vol. 13, no. 8, Aug. 2006.

[LWR12] R. Lyon, T. Walters, and D. Ross, "Intervalgram representation of audio for melody recognition," Patent WO2012/005 970 A2, January, 2012.

[Mar06] M. Marolt, "A mid-level melody-based representation for calculating audio similarity," in *International Symposium on Music Information Retrieval*, Victoria, Canada, Oct. 2006.

[MBHF12] B. Martin, D. G. Brown, P. Hanna, and P. Ferraro, "BLAST for Audio Sequences Alignment: A Fast Scalable Cover Identification Tool," in *International Symposium on Music Information Retrieval*, Porto, Portugal, Oct. 2012, pp. 529–534.

[MD10] M. Mauch and S. Dixon, "Simultaneous estimation of chords and musical context from audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1280–1289, 2010.

[MGB11] A. Muscariello, G. Gravier, and F. Bimbot, "An efficient method for the unsupervised discovery of signalling motifs in large audio streams," in *International Workshop on Content-Based Multimedia Indexing*, Jun. 2011, pp. 145–150.

[MKC05] M. Müller, F. Kurth, and M. Clausen, "Audio matching via chroma-based statistical features," in *International Symposium on Music Information Retrieval*, London, UK, Sep. 2005, pp. 288–295.

[MO08] R. Miotto and N. Orio, "A Music Identification System Based on Chroma Indexing and Statistical Modeling," in *International Symposium on Music Information Retrieval*, Philadelphia, USA, Sep. 2008, pp. 301–306.

[Mye94] E. Myers, "A sublinear algorithm for approximate keyword searching," *Algorithmica*, vol. 12, no. 4-5, pp. 345–374, 1994.

[MZ93] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[NBy98] G. Navarro and R. Baeza-yates, "A practical q-gram index for text retrieval allowing errors," *CLEI Electronic Journal*, vol. 1, 1998.

[NBY00] G. Navarro and R. Baeza-Yates, "A hybrid indexing method for approximate string matching," *Journal of Discrete Algorithms*, vol. 1, no. 1, pp. 205–239, 2000.

[NBySST00] G. Navarro, R. Baeza-yates, E. Sutinen, and J. Tarhio, "Indexing methods for approximate string matching," *IEEE Data Engineering Bulletin*, vol. 24, p. 2001, 2000.

[NKM02] H. Nagano, K. Kashino, and H. Murase, "Fast music retrieval using polyphonic binary feature vectors," in *International Conference on Multimedia and Expo*, vol. 1, Lausanne, Switzerland, Aug. 2002, pp. 101–104 vol.1.

[OE07] J. Ogle and D. Ellis, "Fingerprinting to identify repeated sound events in long-duration personal audio recordings," in *International Conference on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, USA, Apr. 2007.

[OFG11] L. Oudre, C. Févotte, and Y. Grenier, "Probabilistic template-based chord recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2249–2259, 2011.

[OGF09] L. Oudre, Y. Grenier, and C. Févotte, "Chord recognition using measures of fit, chord templates and filtering methods," in *Workshop on the Applications of Signal Processing to Audio and Acoustics*, New York, USA, Oct. 2009, pp. 9–12.

[Ora12] Oracle, "Berkeley DB," Nov. 2012. [Online]. Available: http://www.oracle.com/technetwork/products/berkeleydb/overview/index-085366.html

[OSM04] H. Özer, B. Sankur, and N. Memon, "Robust audio hashing for audio identification," in *European Signal Processing Conference*, vol. 3, Vienna, Austria, Sep. 2004.

[PAO04] J. Pinquier and R. André-Obrecht, "Jingle Detection and Identification in Audio Documents," in *International Conference on Acoustics, Speech and Signal Processing*, vol. 4, Montreal, Canada, May 2004, pp. 329–332.

[Pee07] G. Peeters, "Template-based estimation of time-varying tempo," *EURASIP Journal of Applied Signal Processing*, vol. 2007, no. 1, p. 158, Jan. 2007.

[PP08] H. Papadopoulos and G. Peeters, "Simultaneous estimation of chord progression and downbeats from an audio file," in *International Conference on Acoustics, Speech and Signal Processing*, Las Vegas,USA, Apr. 2008, pp. 121–124.

[RFB+12] M. Ramona, S. Fenet, R. Blouet, H. Bredin, T. Fillon, and G. Peeters, "Audio Fingerprinting: a Public Evaluation Framework Based on a Broadcast Scenario," *Applied Artificial Intelligence: An International Journal*, vol. 26, no. 1-2, pp. 119–136, February 2012.

[Rig85] J. S. Rigden, *Physics and the Sound of Music*. John Wiley & Sons Inc., 1985.

[RP11] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of Bark-bands energy and synchronization through onset detection," in *International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011, pp. 477–480.

[RP13] ——, "AudioPrint: an efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[Sai06] K. Sailer, C.and Rosenbauer, "A bottom-up approach to chord detection," in *International Computer Music Conference*, San Francisco, USA, 2006, pp. 612–615.

[Sch98] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[SE03] A. Sheh and D. Ellis, "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models," in *International Symposium on Music Information Retrieval*, Baltimore, MD, USA, Oct. 2003, p. 185–191.

[Sel80] P. H. Sellers, "The theory and computation of evolutionary distances: Pattern recognition," *Journal of Algorithms*, vol. 1, no. 4, pp. 359 – 373, 1980.

[SGHS08] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, pp. 1138–1151, Aug. 2008.

[SGM11] H. Schreiber, P. Grosche, and M. Müller, "A re-ordering strategy for accelerating index-based audio fingerprinting," in *International Symposium on Music Information Retrieval*, Miami, Florida, USA, October 24-28 2011, pp. 127–132.

[Shi96] F. Shi, "Fast approximate string matching with q-block sequences," in *South American Workshop on String Processing*, Recife, Brazil, Aug. 1996, pp. 257–271.

[Tok12] M. Tokarev, "TinyCDB - a Constant DataBase," May 2012. [Online]. Available: http://www.corpit.ru/mjt/tinycdb.html

[Wan03] A. Wang, "An Industrial-strength Audio Search Algorithm," in *International Symposium on Music Information Retrieval*, Baltimore, Maryland, USA, Oct. 2003, pp. 7 – 13.

[WF74] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974.

[Wit73] F. J. Witty, "The beginnings of indexing and abstracting: Some notes towards a history of indexing and abstracting in antiquity and the middle ages," *The Indexer*, vol. 8, no. 4, pp. 193–198, Oct. 1973.

[WM07] E. Weinstein and P. Moreno, "Music identification with weighted finite-state transducers," in *International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, Apr. 2007, pp. 689–692.

Empreintes Audio et Stratégies d'Indexation Associées pour l'Identification Audio à Grande Echelle

Audio-Fingerprints and Associated Indexing Strategies for the Purpose of Large-Scale Audio-Identification

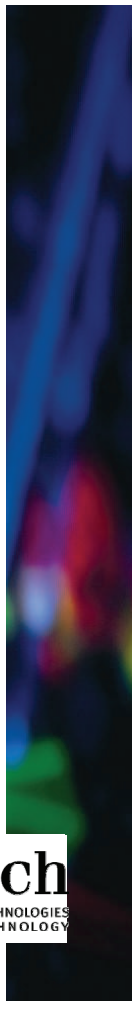
Sébastien FENET

RESUME : Dans cet ouvrage, nous définissons précisément ce qu'est l'identification audio à grande échelle. En particulier, nous faisons une distinction entre l'identification exacte, destinée à rapprocher deux extraits sonores provenant d'un même enregistrement, et l'identification approchée, qui gère également la similarité musicale entre les signaux. A la lumière de ces définitions, nous concevons et examinons plusieurs modèles d'empreinte audio et évaluons leurs performances, tant en identification exacte qu'en identification approchée.

MOTS-CLEFS : identification, extraction d'empreintes, indexation, apprentissage, fouille de données.

ABSTRACT : In this work we give a precise definition of large scale audio identification. In particular, we make a distinction between exact and approximate matching. In the first case, the goal is to match two signals coming from one same recording with different post-processings. In the second case, the goal is to match two signals that are musically similar. In light of these definitions, we conceive and evaluate different audio-fingerprint models.

KEY-WORDS : identification, audio features, indexing, machine learning, data mining.



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY